SkillFlow: Flow-Driven Recursive Skill Evolution for Agentic Orchestration

Mingda Zhang[1], **Tiesunlong Shen**[2], **Haoran Luo**[3], **Wenjin Liu**[3]
Zikai Xiao[4], Erik Cambria[3], Xiaoying Tang[1]

[1]The Chinese University of Hong Kong, Shenzhen [2]National University of Singapore
[3]Nanyang Technological University [4]Zhejiang University

Abstract

In recent years, a variety of powerful LLM-based agentic systems have been applied to automate complex tasks through task orchestration. However, existing orchestration methods still face key challenges, including strategy collapse under reward maximization, high gradient variance with opaque credit assignment, and unguided skill evolution whose decisions are typically made by directly prompting an LLM to judge rather than derived from principled training signals. To address these challenges, we propose SkillFlow, a flow-based framework that takes a trainable Supervisor as the agent and a structured environment with dynamic skill library and frozen executor, automating task orchestration through multi-turn interaction. SkillFlow employs Tempered Trajectory Balance (TTB), a regression-based flow-matching loss that samples trajectories proportional to reward, preserving diverse orchestration strategies rather than collapsing to a single mode. The same flow objective yields a jointly learned backward policy that provides transparent per-step credit assignment at zero additional inference cost. Building on these flow diagnostics, a recursive skill evolution mechanism determines *when* to evolve, *what* skills to create or prune, and *where* decision gaps lie—closing the loop from training signal to autonomous capability growth. Experimental results on 14 datasets show that SkillFlow significantly outperforms baselines across question answering, mathematical reasoning, code generation, and real-world interactive decision making tasks. Our code is available at `https://anonymous.4open.science/r/SkillFlow-E850`.

1 Introduction

In recent years, a variety of powerful LLM-based agentic systems have been applied to solve a wide range of complex tasks [Yao et al., 2022b, Hong et al., 2023, Wang et al., 2024b, Dang et al., 2025], gradually moving beyond single-turn question answering toward executable end-to-end task completion.

In this process, task orchestration has become a key bridge from task goals to reproducible execution: by organizing primitive actions and reusable skills into structured trajectories on an action-level orchestration DAG (Fig. 1, left), where each node is an interaction history and each edge is one orchestration action, agents can

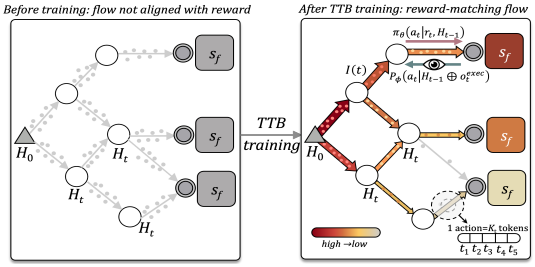


Figure 1: SkillFlow at a glance. **Left**: before training, flow on the orchestration DAG is uniform. **Right**: after TTB, flow concentrates on reward-rich paths (colormap = flow magnitude). **Inset**: each edge is one action (K_t tokens).

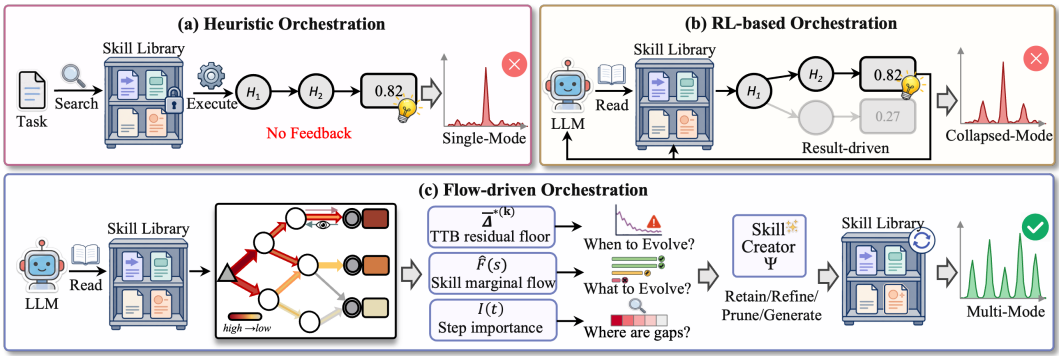


Figure 2: Three orchestration paradigms. (a) heuristic dispatch over a locked library; (b) learning-based with terminal reward, prone to trajectory-level mode collapse; (c) SkillFlow co-trains a forward and backward policy under TTB and uses flow diagnostics to drive recursive skill evolution.

complete complex tasks with improved controllability, compositionality, and reusability [Zeng et al., 2024, Qian et al., 2024]. However, in practice, orchestration still heavily relies on manual skill design and rigid action definitions [Hu et al., 2024, Xu and Yan, 2026], making it costly to transfer across new tasks, new skill configurations, or different agent capabilities.

To address these issues, early approaches rely on heuristic orchestration (Fig. 2a), retrieving skills from prebuilt libraries or searching over workflow graphs at test time, e.g., matching a corresponding tool to each subtask by description, or using tree search to explore multi-step action compositions [Wang et al., 2023, 2024a, Zhang et al., 2024, Zhuge et al., 2024], but their orchestration policies remain bound to pre-designed heuristics and fixed skill repertoires, with no mechanism to adapt from execution outcomes. Recent work has shifted to *learning-based orchestration* (Fig. 2b) [Zeng et al., 2024, Chen et al., 2023, Shao et al., 2024, Guo et al., 2025, Li et al., 2025b, Zhang et al., 2026b, Kong et al., 2026, Wang et al., 2025b, Xia et al., 2026], which trains orchestration policies directly from task-completion signals—sampling orchestration trajectories, collecting terminal rewards such as answer correctness or task success rate, and updating the orchestration model accordingly.

However, these methods still face three challenges. **(i) Strategy collapse.** REINFORCE-family objectives, which uniformly raise or lower all action probabilities along a trajectory based on a single terminal reward, converge to a single reward-maximizing mode, forfeiting diverse, equally effective strategies [Yu et al., 2025, Guo et al., 2025, Zhang et al., 2026b, Kong et al., 2026, Wang et al., 2025b]. This brittleness is amplified when executor capabilities evolve—once a tool is upgraded, the agent has no suitable fallback strategy [Zhang et al., 2023, Li et al., 2025a]. **(ii) High gradient variance and opaque credit assignment.** Terminal-only rewards create long credit-assignment horizons, where policy-gradient variance and intra-group advantage collapse make per-step attribution opaque, when a multi-step trajectory succeeds or fails, it remains unclear which step is responsible [Schulman et al., 2015, Tan et al., 2026, Wang et al., 2026c, Li et al., 2025b, Wang et al., 2025b]. **(iii) Unguided skill evolution.** Existing frameworks with dynamic skill libraries rely on heuristic triggers, fixed schedules, or direct LLM-as-judge prompting, lacking a principled signal for *when* to update the library, *what* skills to prune or create, or *where* the critical decision points lie—so the library evolves blindly [Wang et al., 2023, Xia et al., 2026, Fang et al., 2025, Xu and Yan, 2026, Alzubi et al., 2026, Zhang et al., 2026a, Wang et al., 2025a, 2026a].

To address these challenges, we propose **SkillFlow** (Fig. 2c), a flow-based framework [Bengio et al., 2021, 2023, Malkin et al., 2022] for general task orchestration with recursive skill evolution. A trainable Supervisor interacts with a structured environment containing a dynamic skill library and a frozen executor. At the core of SkillFlow (Fig. 1) is **Tempered Trajectory Balance (TTB)**, a regression-style flow-matching loss that drives each trajectory's sampling probability to be *proportional to its reward*, rather than concentrated on a single best outcome. In contrast, REINFORCE-family objectives push nearly all probability mass onto one reward-maximizing trajectory; this reward-proportional sampling instead keeps multiple high-reward sub-trajectories—distinct successful paths through the orchestration DAG—alive under a single loss, preserving strategic diversity. The same loss jointly trains a *backward policy* that, once an orchestration terminates, attributes the trajectory-level outcome to its individual steps at zero additional inference cost—flagging which decisions actually drove success and which were incidental. Building on these per-step and per-skill credit signals, a recursive

skill evolution mechanism answers three questions directly from the training signal itself: *when* the current library starts limiting performance (signaled by TTB convergence), *what* skills to create or prune (ranked by the skill marginal flow $\hat{F}(s)$), and *where* decision gaps lie (localized by the step importance $I(t)$)—closing the loop from training signal to autonomous capability growth.

We evaluate on benchmarks across question answering, mathematical reasoning, interactive decision making, and code generation. Results show SkillFlow outperforms direct LLMs, REINFORCE-style RL baselines [Li et al., 2025b, Zhang et al., 2026b, Xia et al., 2026], and skill-evolution methods [Jiang et al., 2026a, Yang et al., 2026b, Ma et al., 2026, Alzubi et al., 2026] in task accuracy, strategy diversity, and orchestration cost—a foundation for self-evolving, flow-based agentic orchestration.

2 Related Work

Agent Task Orchestration. LLM-era task orchestration has evolved from rule-based automation to feedback-driven plan–act–feedback loops [Yao et al., 2022b, Wang et al., 2024a,b]. Existing work spans single-agent sequential decision making [Qin et al., 2023, Yao et al., 2022b], LLM-controller-based routing and constrained API planning [Ong et al., 2024, Wang et al., 2024a, Su et al., 2026], and multi-agent SOP/role collaboration [Hong et al., 2023, Wu et al., 2024, Dang et al., 2025]. Reusable skill memory [Wang et al., 2023, Xu and Yan, 2026, Jiang et al., 2026b, Wang et al., 2026a] and self-evolving architectures [Fang et al., 2025, Li, 2026, Alzubi et al., 2026, Zhang et al., 2026a, Wang et al., 2025a, 2026b] further improve efficiency, yet all these approaches operate with fixed skill sets and lack a principled mechanism to expand the action space during training. SkillFlow fills this gap by learning orchestration from execution feedback while autonomously evolving its skill repertoire.

Reinforcement Learning for Agents. Agent RL models multi-turn interaction as a long-horizon MDP [Zhou et al., 2023, Chen et al., 2025]. Credit-assignment advances include implicit step rewards [Liu et al., 2025], hindsight advantage estimation [Tan et al., 2026], plan-execute decomposition [Peng et al., 2026], progressive reward shaping [Su et al., 2025], and process reward models [Zhang et al., 2025, She et al., 2025]. Policy optimization has converged on GRPO-style objectives [Guo et al., 2025, Shao et al., 2024] with extensions (DAPO [Yu et al., 2025], VAPO [Yue et al., 2025], multi-agent variants [Liu et al., 2026, Cang et al., 2026]), also applied to workflow orchestration [Zhang et al., 2026b, Li et al., 2025b] and search-augmented reasoning [Jin et al., 2025, Yang et al., 2026a]. All rely on REINFORCE-family objectives that converge to a single mode, lacking diversity-preserving signals [Li et al., 2025a]. While reward-matching flow training has been explored in LLM reasoning [Yu et al., 2024] and robust scheduling [Zhang et al., 2023], SkillFlow targets a different setting—multi-turn agentic orchestration with a co-evolving skill library—and contributes reward-proportional trajectory sampling together with zero-cost per-step credit.

3 Preliminaries

Definition 1: Orchestration State Graph. We model the task orchestration process as a directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where each vertex $H_t \in \mathcal{V}$ represents an interaction history and each edge $(H_{t-1}, H_t) \in \mathcal{A}$ corresponds to an orchestration action. Acyclicity follows from the state update rule $H_t = H_{t-1} \oplus (r_t, a_t, o_t^{\text{exec}})$, which yields strict history growth $|H_t| > |H_{t-1}|$.

Definition 2: Orchestration Trajectory. A complete path through $\mathcal{G}$ from initial state H_0 to a terminal state defines an orchestration trajectory:

$$\tau = \left\{(r_t,\ a_t,\ o_t^{\text{exec}})\right\}_{t=1}^{T} \ \Rightarrow\ y_q, \tag{1}$$

where r_t denotes the reasoning reflection at step t, $a_t = (\alpha_t, o_t)$ is the action with type $\alpha_t \in \{\texttt{skill}, \texttt{act}, \texttt{accept}\}$ and parameters o_t, and o_t^{exec} is the execution feedback. The episode terminates when $\alpha_t = \texttt{accept}$ or $t = T_{\max}$.

Problem Statement. Given task $q \in \mathcal{D}$, environment $\mathcal{E}$, and executor $\mathcal{M}_{\text{exec}}$, we augment $\mathcal{G}$ with a non-negative flow function $F : \mathcal{V} \to \mathbb{R}_{\geq 0}$ satisfying conservation (incoming flow equals outgoing flow at each non-terminal state), with terminal condition $F(x) = \tilde{R}(\tau_x)^\beta$. The resulting flow network [Bengio et al., 2021, 2023] (formal foundations in Appendix A) induces a policy that samples trajectories in proportion to reward:

$$\pi^*(\tau \mid q) \propto \tilde{R}(\tau)^\beta, \quad \tilde{R}(\tau) = R(\tau) + \varepsilon_{\min}, \tag{2}$$

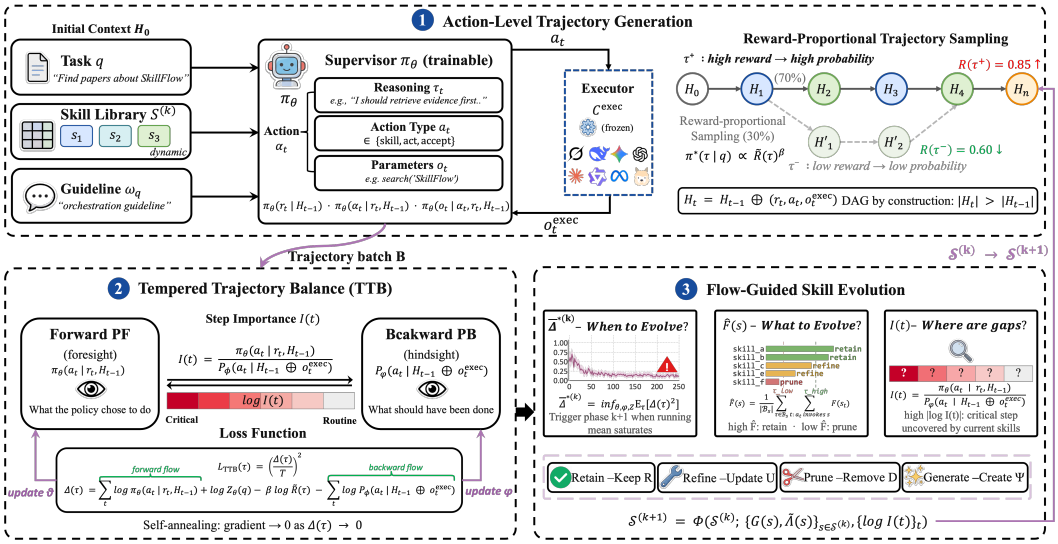


Figure 3: SkillFlow architecture. The Supervisor rolls out a tree-structured DAG against a frozen Executor; TTB jointly trains the forward and hindsight backward policies via a flow-matching residual loss; flow diagnostics ($\bar{\Delta}^{*(k)}$, $\hat{F}(s)$, $I(t)$) drive recursive skill curation at phase boundaries.

where $R(\tau)$ is task completion quality (per-task definitions in Appendix G), $\varepsilon_{\min} > 0$ is a small constant that shifts rewards to be strictly positive—a prerequisite of flow networks—so that even zero-reward trajectories receive non-zero flow (details in Appendix H), and $\beta > 0$ controls the diversity–quality tradeoff. This is equivalent to entropy-regularized RL with temperature $T = 1/\beta$ (Appendix C).

4 Methodology: SkillFlow

As illustrated in Figure 3, this section introduces the SkillFlow framework, including environment design and task modeling (Section 4.1), flow-based end-to-end training (Section 4.2), and flow-driven recursive skill evolution (Section 4.3).

4.1 Environment Design and Task Modeling

SkillFlow follows a Supervisor-Executor paradigm [Yao et al., 2022b]: a trainable Supervisor π_θ interacts with the structured environment $\mathcal{E}$ to construct orchestration trajectories. Unlike static frameworks that operate on a fixed skill set, SkillFlow's environment carries a **dynamic skill library** that co-evolves with the policy through the curation operator Φ formalised in §4.3.

Structured Environment $\mathcal{E}$. The environment maintains the skill library, skill creator, and executor:

$$\mathcal{E} = (\mathcal{S},\ \Psi,\ \mathcal{M}_{\text{exec}}), \tag{3}$$

where $\mathcal{S}$ is the dynamic skill library, Ψ is the Skill Creator that evolves $\mathcal{S}$ via $s_{\text{new}} = \Psi(c, \mathcal{T}, \mathcal{S})$ using creation context c and trajectory evidence $\mathcal{T}$ (Section 4.3), and $\mathcal{M}_{\text{exec}}$ is the pluggable executor. Ψ updates $\mathcal{S}$ only at episode boundaries, keeping the action space constant within each training phase.

State Space $\mathcal{H}$ and Orchestration Target. The initial state concatenates the task with retrieved context: $H_0 = [q \oplus \mathcal{S}_{\text{ret}} \oplus \omega_q]$, where $\mathcal{S}_{\text{ret}}$ are retrieved skills and ω_q is a task-category-specific orchestration guideline. The state evolves via

$$H_t = H_{t-1} \oplus (r_t,\ a_t,\ o_t^{\text{exec}}). \tag{4}$$

The Supervisor interacts with $\mathcal{E}$ until $\alpha_t = \texttt{accept}$ (early termination) or $t = T_{\max}$ (budget exhausted); the resulting trajectory τ determines the answer y_q via the terminal action's output.

Step-wise Orchestration Policy. At each step t, the Supervisor generates reasoning r_t, selects action type α_t, and produces parameters o_t, modeled as a hierarchical policy conditioned on H_{t-1}:

$$\pi_\theta(r_t, a_t \mid H_{t-1}) = \pi_\theta(r_t \mid H_{t-1}) \cdot \pi_\theta(\alpha_t \mid r_t, H_{t-1}) \cdot \pi_\theta(o_t \mid \alpha_t, r_t, H_{t-1}). \tag{5}$$

Multi-Turn Interaction and Trajectory Distribution. Given action a_t, the executor returns feedback $o_t^{\text{exec}} \sim \mathcal{C}_{\text{exec}}(\cdot \mid H_{t-1}, a_t)$, the state evolves via Eq. 4, and the Supervisor continues until termination. Marginalising the joint policy–executor draws over the T steps yields the trajectory distribution:

$$P_\theta(\tau) = \prod_{t=1}^{T} \big[\pi_\theta(r_t, a_t \mid H_{t-1}) \cdot \mathcal{C}_{\text{exec}}(o_t^{\text{exec}} \mid H_{t-1}, a_t)\big], \tag{6}$$

where only π_θ is trainable and $\mathcal{C}_{\text{exec}}$ is frozen.

Proposition 1. *The Supervisor–Executor environment admits a flow-conservative DAG structure suitable for end-to-end flow-based training.*

Proof. Main-result gains (§5.2, RQ1) and cross-backbone transferability (§5.4, RQ3) provide empirical validation; formal acyclicity in Appendix E. □

4.2 Flow-Based End-to-End Training

Building on the DAG structure of $\mathcal{G}$ (Proposition 1), we train the flow network introduced in §3 to make the action-sequence distribution reward-proportional given the realized reasoning and execution context: $\pi_\theta(a_{1:T} \mid r_{1:T}, o_{1:T}^{\text{exec}}, q) \propto \tilde{R}(\tau)^\beta$.

Forward Policy P_F and Backward Policy P_ϕ. Because $\mathcal{C}_{\text{exec}}$ is frozen and reasoning r_t is fixed context, the forward policy reduces to action selection, while the backward policy P_ϕ conditions on the **hindsight state** $H_{t-1} \oplus o_t^{\text{exec}}$ to incorporate the execution observation unavailable to π_θ:

$$P_F(H_t \,|\, H_{t-1}) = \pi_\theta(a_t \,|\, r_t, H_{t-1}), \quad P_B(H_{t-1} \,|\, H_t) = P_\phi\left(a_t \,\middle|\, \underbrace{H_{t-1} \oplus o_t^{\text{exec}}}_{\text{hindsight state}}\right). \tag{7}$$

Tempered Trajectory Balance (TTB). Given P_F and P_ϕ, the (tempered, hindsight-conditioned) **Trajectory Balance** (TB) condition [Bengio et al., 2023] requires for every trajectory:

$$\log Z_\theta(q) + \sum_{t=1}^{T} \log P_F(H_t \,|\, H_{t-1}) = \beta \log \tilde{R}(\tau) + \sum_{t=1}^{T} \log P_B(H_{t-1} \,|\, H_t). \tag{8}$$

The **TTB loss** is the squared, length-normalized residual of this condition [Dall'Antonia et al., 2026] (full derivation in Appendix B):

$$\Delta(\tau) \coloneqq \log Z_\theta(q) + \sum_{t=1}^{T} \log \pi_\theta(a_t \mid r_t, H_{t-1}) - \beta \log \tilde{R}(\tau) - \sum_{t=1}^{T} \log P_\phi\big(a_t \mid H_{t-1} \oplus o_t^{\text{exec}}\big),$$

$$\mathcal{L}_{\text{TTB}}(\tau) = \big(\Delta(\tau)/T\big)^2. \tag{9}$$

Here $T = |\tau|$, $Z_\theta(q)$ is a task-conditioned partition function, and each per-step log-probability is *per-token normalized* (details in Appendix B.3). At the optimum $\Delta(\tau) = 0$, the conditional action-sequence distribution becomes reward-proportional, $\pi_\theta(a_{1:T} \mid r_{1:T}, o_{1:T}^{\text{exec}}, q) \propto \tilde{R}(\tau)^\beta$.

Step Importance and Skill Marginal Flow. Each H_t has a unique parent; $\mathcal{G}$ is therefore tree-structured, and TB convergence implies Detailed Balance $F(H)\, P_F(H'|H) = F(H')\, P_B(H|H')$ at every edge (formal treatment in Appendix D). Rearranging yields the **step importance**:

$$I(t) = \frac{F(H_t)}{F(H_{t-1})} = \frac{\pi_\theta(a_t \mid r_t, H_{t-1})}{P_\phi(a_t \mid H_{t-1} \oplus o_t^{\text{exec}})}, \tag{10}$$

which incurs no extra inference cost. The **information asymmetry**—π_θ decides without o_t^{exec}, while P_ϕ evaluates with it—makes $I(t)$ a credit signal: large $|\log I(t)|$ marks decisions whose appraisal shifted after execution. Telescoping from $F(s_0) = Z_\theta(q)$ gives the **skill marginal flow**:

$$\hat{F}(s) = \frac{1}{|\mathcal{B}_s|} \sum_{\tau \in \mathcal{B}_s} \sum_{t:\, a_t \text{ invokes } s} F(H_t), \qquad \log F(H_t) = \log Z_\theta(q) + \sum_{t'=1}^{t} \log I(t'), \tag{11}$$

where $\mathcal{B}_s \subseteq \mathcal{B}$ is the subset of trajectories invoking s and the inner sum aggregates flow over each occurrence of s within a trajectory.

Proposition 2. *TTB training induces reward-proportional sampling and yields per-step credit at no extra inference cost.*

Proof. Main results (§5.2, RQ1), OOD (§5.3, RQ2), −TTB ablation (§5.5, RQ4), and algorithm comparison (§5.6, RQ5) provide empirical validation; full proof in Appendix J. □

4.3 Flow-Driven Recursive Skill Evolution

Prior work on skill distillation and refinement [Wang et al., 2023, Alzubi et al., 2026, Zhang et al., 2026a, Wang et al., 2026b, Xia et al., 2026] relies on heuristic schedules or LLM-as-judge prompting, leaving *when*, *what*, and *where* to evolve underspecified. SkillFlow derives all three from the flow signals of Section 4.2: the TTB residual $\Delta(\tau)$ signals *when*, while the step importance $I(t)$ and skill marginal flow $\hat{F}(s)$ localize *what* and *where*.

When: TTB Residual Floor. Within phase k, gradient descent drives $\Delta(\tau)^2 \to 0$ (Proposition 2). The squared-residual floor under the current library is

$$\bar{\Delta}^{*(k)} \;\coloneqq\; \inf_{\theta,\phi,Z_\theta} \mathbb{E}_\tau\big[\Delta(\tau \mid \mathcal{S}^{(k)}, \theta, \phi, Z_\theta)^2\big] \;\geq\; 0. \tag{12}$$

$\bar{\Delta}^{*(k)} = 0$ when $\mathcal{S}^{(k)}$ together with the policy class can express the reward-proportional flow; otherwise the loss plateaus at $\bar{\Delta}^{*(k)} > 0$. Phase $k{+}1$ is triggered when the running mean of $\Delta(\tau)^2$ saturates against this plateau.

What and Where: Flow-Guided Skill Curation via a CGF. Each skill $s \in \mathcal{S}$ is an *atomic tip*—a short, self-contained piece of strategic guidance, independently composable via the `skill` action. We define the **per-skill cumulant generating function** (CGF) of the telescoped log-flow:

$$\Lambda_\lambda^{(s)} \;\coloneqq\; \log\left(\frac{1}{|\mathcal{B}_s|}\sum_{\tau\in\mathcal{B}_s}\sum_{t:\,a_t \text{ invokes } s}\exp\Big(\lambda\sum_{t'=1}^{t}\log I(t')\Big)\right), \quad \lambda\in\mathbb{R} \text{ (moment order)},$$

which, by the telescoping identity $\log F(H_t) = \log Z_\theta(q) + \sum_{t'\leq t}\log I(t')$ (Eq. 11), is the log λ-th moment of $F(H_t)/Z_\theta(q)$ along occurrences of s. $\Lambda_\lambda^{(s)}$ is convex in λ; two summaries derived from it drive library evolution—the **mean log-flow** $G(s) \coloneqq \frac{\partial\Lambda_\lambda^{(s)}}{\partial\lambda}\big|_{\lambda=0}$ measures the average flow s attracts when invoked, and the **centered log-flow share** $\widetilde{\Lambda}(s) \coloneqq \Lambda_1^{(s)} - \mathbb{E}_{s'}[\Lambda_1^{(s')}]$ ranks s's marginal contribution to the library's reward-proportional sampling:

$$\mathcal{S}^{(k+1)} \;=\; \Phi\big(\mathcal{S}^{(k)};\, \{G(s),\, \widetilde{\Lambda}(s)\}_{s\in\mathcal{S}^{(k)}},\, \{\log I(t)\}_t\big), \tag{13}$$

where $\Lambda_1^{(s)} = \log\hat{F}(s) - \log Z_\theta(q)$ recovers the skill marginal flow in log-space. The operator Φ partitions $\mathcal{S}^{(k)}$ into four disjoint classes: *retain* (high $G(s)$, small Jensen gap), *refine* (high $G(s)$, persistently large Jensen gap), and *prune* (persistently negative $\widetilde{\Lambda}(s)$); on top of these it invokes the Skill Creator Ψ to *generate* new atomic tips at high-$\log I(t)$ steps from same-query success/failure pairs (τ^+, τ^-). The Jensen gap $\Lambda_1^{(s)} - G(s)$ equals the cross-visit variance of $\log F(H_t)$ plus higher-order cumulants, serving as a stability diagnostic that distinguishes *retain* from *refine* for context-inconsistent skills.

Proposition 3. *Flow-driven recursive evolution autonomously expands the skill library while preserving its atomic composability.*

Proof. Leave-one-out ablation (§5.5, RQ4) and skill-evolution cost savings (§5.6, RQ5) provide empirical validation; full proof in Appendix K. □

5 Experiments

We evaluate SkillFlow through the following research questions (RQs): **RQ1:** Can SkillFlow outperform existing workflow orchestration methods on in-distribution benchmarks? **RQ2:** How does SkillFlow generalize to out-of-distribution benchmarks? **RQ3:** How transferable is SkillFlow across different LLM backbones? **RQ4:** What are the contributions of core components such as TTB training, backward policy, and skill evolution? **RQ5:** How does SkillFlow compare against other agent RL algorithms (GRPO, Tree-GRPO, HCAPO) and skill-evolution baselines in accuracy and computational cost?

		Baseline		SFT	GRPO	AFlow	Agent+RL			Ours
Dataset	**Metric**	**Qwen3.5**	**v4-flash**	**Qwen3.5**	**Qwen3.5**	**Qwen3.5**	**AgentFlow**	**FlowSteer**	**SkillRL**	**SkillFlow (Δ ↑)**
(a) In-Distribution (IID) benchmarks										
HotpotQA	Ans EM	$60.94_{\pm 0.8}$	$72.66_{\pm 0.1}$	$64.84_{\pm 0.4}$	$69.53_{\pm 0.3}$	$88.28_{\pm 0.9}$	$88.28_{\pm 0.8}$	$89.84_{\pm 1.1}$	$87.50_{\pm 0.2}$	**92.19** (+31.3)
	Ans F1	$75.70_{\pm 0.1}$	$85.95_{\pm 0.3}$	$79.08_{\pm 0.7}$	$81.52_{\pm 0.1}$	$90.92_{\pm 0.3}$	$90.11_{\pm 0.8}$	$91.20_{\pm 0.7}$	$89.16_{\pm 0.3}$	**93.95** (+18.3)
TriviaQA	Ans EM	$44.88_{\pm 1.0}$	$74.02_{\pm 0.1}$	$47.24_{\pm 1.0}$	$47.24_{\pm 0.9}$	$92.97_{\pm 0.5}$	$89.84_{\pm 0.3}$	$91.41_{\pm 1.2}$	$89.06_{\pm 0.5}$	**96.09** (+51.2)
	Ans F1	$54.30_{\pm 0.2}$	$81.99_{\pm 1.0}$	$55.73_{\pm 0.8}$	$56.93_{\pm 1.0}$	$93.25_{\pm 0.9}$	$90.47_{\pm 0.7}$	$92.06_{\pm 1.2}$	$90.63_{\pm 0.5}$	**96.94** (+42.6)
AIME 2026	Acc.	$46.67_{\pm 1.0}$	$50.00_{\pm 0.8}$	$36.67_{\pm 1.0}$	$33.33_{\pm 0.7}$	$53.33_{\pm 0.9}$	$60.00_{\pm 0.2}$	$63.33_{\pm 0.4}$	$56.67_{\pm 0.4}$	**70.00** (+23.3)
MedQA	Acc.	$69.53_{\pm 0.4}$	$83.59_{\pm 0.2}$	$73.44_{\pm 0.4}$	$77.34_{\pm 0.8}$	$89.84_{\pm 0.5}$	$87.50_{\pm 0.5}$	$90.63_{\pm 0.3}$	$85.94_{\pm 0.4}$	**92.19** (+22.7)
WebShop	Avg Score	$56.93_{\pm 0.8}$	$85.17_{\pm 0.8}$	$50.80_{\pm 0.3}$	$54.10_{\pm 0.9}$	$71.09_{\pm 0.3}$	$83.61_{\pm 0.5}$	$85.43_{\pm 1.2}$	$87.96_{\pm 0.8}$	**94.73** (+37.8)
	SR	$32.03_{\pm 0.9}$	$67.97_{\pm 1.0}$	$35.16_{\pm 1.0}$	$36.71_{\pm 0.4}$	$59.38_{\pm 0.1}$	$74.22_{\pm 0.4}$	$79.20_{\pm 0.4}$	$82.81_{\pm 0.3}$	**93.75** (+61.7)
ALFWorld	SR	$48.28_{\pm 1.1}$	$61.21_{\pm 0.4}$	$40.52_{\pm 0.8}$	$50.78_{\pm 0.5}$	$75.00_{\pm 1.1}$	$80.47_{\pm 0.6}$	$82.81_{\pm 0.4}$	$85.16_{\pm 0.4}$	**96.09** (+47.8)
SWE-bench	Resolved	$17.19_{\pm 0.4}$	$38.28_{\pm 0.7}$	$16.41_{\pm 1.1}$	$18.75_{\pm 0.5}$	$30.47_{\pm 0.3}$	$41.41_{\pm 1.2}$	$43.75_{\pm 0.7}$	$39.06_{\pm 0.2}$	**52.34** (+35.2)
	Ans EM	$52.91_{\pm 0.9}$	$73.34_{\pm 0.1}$	$56.04_{\pm 0.7}$	$58.39_{\pm 0.6}$	$90.63_{\pm 0.7}$	$89.06_{\pm 0.6}$	$90.63_{\pm 1.2}$	$88.28_{\pm 0.4}$	**94.14** (+41.2)
Avg. (IID)	Ans F1	$65.00_{\pm 0.2}$	$83.97_{\pm 0.7}$	$67.41_{\pm 0.8}$	$69.23_{\pm 0.6}$	$92.09_{\pm 0.6}$	$90.29_{\pm 0.8}$	$91.63_{\pm 1.0}$	$89.90_{\pm 0.4}$	**95.45** (+30.5)
	Acc./Pass	$45.11_{\pm 0.8}$	$64.37_{\pm 0.7}$	$42.17_{\pm 0.8}$	$45.17_{\pm 0.6}$	$63.19_{\pm 0.5}$	$71.20_{\pm 0.6}$	$74.19_{\pm 0.6}$	$72.93_{\pm 0.4}$	**83.18** (+38.1)
(b) Out-of-Distribution (OOD) benchmarks										
MuSiQue	Ans EM	$39.06_{\pm 0.2}$	$50.78_{\pm 0.8}$	$39.10_{\pm 1.0}$	$42.20_{\pm 0.6}$	$78.13_{\pm 0.2}$	$79.69_{\pm 0.5}$	$79.69_{\pm 1.2}$	$75.78_{\pm 0.7}$	**85.16** (+46.1)
	Ans F1	$47.79_{\pm 1.0}$	$59.32_{\pm 0.1}$	$47.90_{\pm 0.9}$	$50.30_{\pm 0.8}$	$84.38_{\pm 0.7}$	$84.90_{\pm 0.4}$	$85.22_{\pm 0.8}$	$81.38_{\pm 0.2}$	**89.29** (+41.5)
NQ-Open	Ans EM	$21.88_{\pm 0.6}$	$37.50_{\pm 1.1}$	$23.44_{\pm 1.1}$	$21.88_{\pm 0.4}$	$75.79_{\pm 0.7}$	$78.91_{\pm 0.3}$	$80.47_{\pm 1.1}$	$78.91_{\pm 1.1}$	**82.81** (+60.9)
	Ans F1	$30.03_{\pm 0.8}$	$47.90_{\pm 0.8}$	$30.97_{\pm 0.3}$	$28.84_{\pm 0.9}$	$80.23_{\pm 0.7}$	$83.44_{\pm 1.0}$	$84.88_{\pm 0.7}$	$83.09_{\pm 0.1}$	**86.04** (+56.0)
MATH-Hard	Acc.	$89.06_{\pm 0.1}$	$91.41_{\pm 1.1}$	$88.28_{\pm 1.1}$	$87.50_{\pm 1.0}$	$90.63_{\pm 0.4}$	$92.19_{\pm 0.2}$	$93.75_{\pm 1.1}$	$91.41_{\pm 1.1}$	**96.09** (+7.0)
GPQA Diamond	Acc.	$61.72_{\pm 0.6}$	$73.44_{\pm 0.2}$	$71.88_{\pm 0.9}$	$75.00_{\pm 0.9}$	$75.78_{\pm 0.2}$	$81.25_{\pm 0.6}$	$84.38_{\pm 0.7}$	$78.13_{\pm 0.4}$	**89.84** (+28.1)
HumanEval	pass@1	$89.06_{\pm 0.6}$	$96.88_{\pm 0.3}$	$80.47_{\pm 0.7}$	$85.94_{\pm 0.9}$	$92.97_{\pm 0.3}$	$93.75_{\pm 0.4}$	$93.75_{\pm 1.2}$	$92.19_{\pm 0.8}$	**98.44** (+9.4)
ScienceWorld	Success	$25.78_{\pm 0.7}$	$43.43_{\pm 0.2}$	$15.62_{\pm 0.3}$	$24.22_{\pm 0.5}$	$29.69_{\pm 0.7}$	$44.53_{\pm 0.4}$	$47.66_{\pm 0.3}$	$39.06_{\pm 0.2}$	**57.81** (+32.0)
	Avg Score	$41.86_{\pm 0.4}$	$58.64_{\pm 1.1}$	$34.69_{\pm 1.0}$	$40.00_{\pm 0.2}$	$43.65_{\pm 0.4}$	$56.85_{\pm 0.8}$	$58.91_{\pm 0.3}$	$50.11_{\pm 0.2}$	**67.87** (+26.0)
Mind2Web	Step Acc	$26.64_{\pm 0.7}$	$27.57_{\pm 0.6}$	$18.74_{\pm 1.0}$	$23.51_{\pm 1.0}$	$30.47_{\pm 0.3}$	$39.84_{\pm 0.2}$	$41.41_{\pm 0.6}$	$35.16_{\pm 0.6}$	**51.56** (+24.9)
	Action F1	$63.85_{\pm 0.9}$	$66.21_{\pm 0.8}$	$50.49_{\pm 1.2}$	$66.98_{\pm 0.2}$	$62.29_{\pm 0.5}$	$71.77_{\pm 0.5}$	$75.22_{\pm 1.0}$	$69.54_{\pm 0.4}$	**84.89** (+21.0)
	EM	$30.47_{\pm 0.4}$	$44.14_{\pm 1.0}$	$31.27_{\pm 1.1}$	$32.04_{\pm 0.5}$	$76.96_{\pm 0.5}$	$79.30_{\pm 0.4}$	$80.08_{\pm 1.2}$	$77.34_{\pm 0.9}$	**83.99** (+53.5)
Avg. (OOD)	F1	$47.22_{\pm 0.9}$	$57.81_{\pm 0.6}$	$43.12_{\pm 0.8}$	$48.71_{\pm 0.6}$	$75.63_{\pm 0.6}$	$80.04_{\pm 0.6}$	$81.77_{\pm 0.8}$	$78.00_{\pm 0.2}$	**86.74** (+39.5)
	Acc./Pass	$55.69_{\pm 0.5}$	$65.23_{\pm 0.6}$	$51.61_{\pm 0.8}$	$56.03_{\pm 0.8}$	$60.53_{\pm 0.4}$	$68.07_{\pm 0.4}$	$69.98_{\pm 0.7}$	$64.34_{\pm 0.6}$	**76.94** (+21.3)

Table 1: Main results on 14 IID and OOD benchmarks. SkillFlow uses Qwen3.5-9B as the Supervisor. "Agent+RL" covers AgentFlow, FlowSteer, SkillRL. $\Delta \uparrow$ over Qwen3.5-9B.

5.1 Experimental Setup

Datasets. We evaluate SkillFlow on 14 benchmarks covering four task categories. *In-distribution (IID, 7):* HotpotQA [Yang et al., 2018], TriviaQA [Joshi et al., 2017], MedQA [Jin et al., 2021], AIME 2026, WebShop [Yao et al., 2022a], ALFWorld [Shridhar et al., 2020], SWE-bench [Jimenez et al., 2023]. *Out-of-distribution (OOD, 7):* MuSiQue [Trivedi et al., 2022], NQ-Open, MATH-Hard, GPQA Diamond, HumanEval [Chen et al., 2021], ScienceWorld, Mind2Web. More details in Appendix M.

Baselines. We compare against *(i)* direct LLMs (Qwen3.5-9B, v4-flash, Claude Haiku 4.5), *(ii)* fine-tuning (SFT, GRPO [Shao et al., 2024]), *(iii)* search-based workflows (AFlow [Zhang et al., 2024]), and *(iv)* RL agents (AgentFlow [Li et al., 2025b], FlowSteer [Zhang et al., 2026b], SkillRL [Xia et al., 2026]). See Appendix N for details.

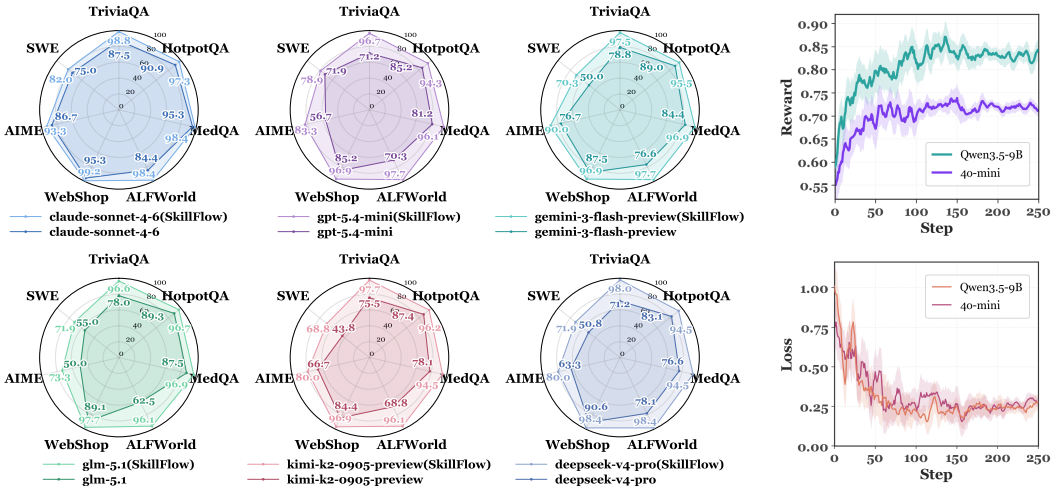
Evaluation Metrics. F1 for QA, Accuracy for AIME/MedQA/MATH/GPQA, Average Score and Success Rate for WebShop/ALFWorld, Resolved Rate for SWE-bench, pass@1 for HumanEval. Details in Appendix O.

5.2 Main Results (RQ1)

Table 1 shows SkillFlow leads across all 14 benchmarks and surpasses even much stronger direct-LLM baselines (v4-flash, Claude Haiku 4.5), indicating gains come from how the orchestration policy is trained rather than from backbone capacity. Margins concentrate on WebShop, ALFWorld, and SWE-bench — where REINFORCE-family baselines suffer strategy collapse and AFlow's static workflow runs out of moves — and persist across both IID and OOD halves. The gains compound from three mechanisms: TTB's regression loss has lower variance than REINFORCE (formal bound in Appendix I), the backward policy supplies zero-cost per-step credit, and recursive skill evolution removes the static-library bottleneck of AgentFlow, FlowSteer, and SkillRL.

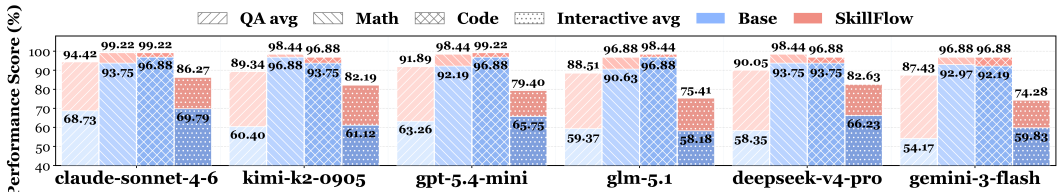
Variant	IID							OOD						
	Hotpot EM	Trivia EM	AIME Acc	MedQA Acc	WShop SR	ALFW SR	SWE Res	MuSQ EM	NQ EM	MATH Acc	GPQA Acc	HEval pass@1	SciW Succ	M2W Step Acc
− TTB	83.59	87.50	53.33	85.16	82.03	82.81	40.63	73.44	79.69	87.50	79.69	89.06	39.10	33.59
− Backward policy	85.94	90.63	53.33	86.72	80.47	88.28	43.75	75.00	78.91	89.06	83.09	90.63	42.20	36.72
− $\dot{\mathcal{L}}_{\text{TTB}}$ for *when*	82.81	84.38	60.00	89.06	83.02	85.93	46.88	75.78	71.88	90.63	84.88	92.19	48.44	40.63
− $I(t)$ for *where*	85.16	88.26	63.33	89.06	87.50	90.63	44.53	78.13	75.00	88.28	82.81	93.75	47.90	42.97
− $\hat{F}(s)$ for *what*	84.38	82.03	50.00	83.59	79.69	80.47	42.20	76.56	75.78	87.50	80.47	90.63	45.31	42.19
SkillFlow (Full)	**92.19**	**96.09**	**70.00**	**92.19**	**93.75**	**96.09**	**52.34**	**85.16**	**82.81**	**96.09**	**89.84**	**98.44**	**57.81**	**51.56**

Table 2: Component ablation across IID and OOD benchmarks (RQ4). Each row removes one component. −*TTB* replaces TTB with GRPO (C1); −*Backward policy* removes the hindsight P_ϕ (C2); the last three rows ablate the *when*/*where*/*what* signals of skill evolution (C3).



(a) Radar charts on different backbones

(b) Training dynamics



(c) Aggregated performance by task type

Figure 4: Backbone transferability. (a) Per-backbone radar across seven IID benchmarks (six proprietary LLMs), with vs. without SkillFlow. (b) Training dynamics on two trainable backbones. (c) Aggregated gain by task category. SkillFlow lifts every backbone, with weaker ones gaining most.

5.3 Out-of-Distribution Generalization (RQ2)

The seven OOD benchmarks (Table 1(b)) share the four task categories with the IID set but contain no training data, and the skill library is frozen at end-of-training — a strict transfer test. SkillFlow retains its IID lead with comparable margins; notably, the gap to REINFORCE-style baselines *widens* on OOD because reward-proportional sampling preserves multiple solution paths, whereas a single-mode policy fails once surface forms shift. On Avg.(OOD) F1, the lift over FlowSteer reaches $+6.08\%$—roughly $1.5\times$ the IID gap. That a frozen library still helps unseen tasks indicates the evolved skills capture transferable orchestration primitives rather than benchmark-specific shortcuts.

5.4 Backbone Transferability (RQ3)

Figure 4 swaps the Supervisor for six proprietary LLMs. Weaker backbones gain most—explicit credit and diversity-preserving sampling offset shaky base reasoning—while stronger ones still gain on agent-style tasks where orchestration is the bottleneck. The per-category lift hierarchy

(i) IID benchmarks

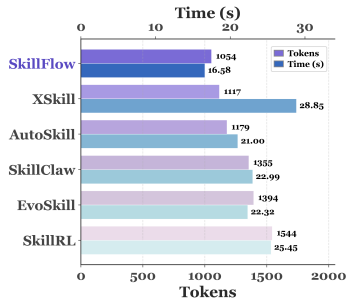
Method	Hotpot EM	Trivia EM	AIME Acc	MedQA Acc	WShop SR	ALFW SR	SWE Res
Qwen3.5	60.94	44.88	46.67	69.53	32.03	48.28	17.19
GRPO	83.59	87.50	53.33	85.16	82.03	82.81	40.63
Tree-GRPO	87.50	93.75	60.00	88.28	85.94	90.92	48.44
HCAPO	85.94	93.75	63.33	90.62	87.50	92.19	47.24
SkillFlow (Ours)	**92.19**	**96.09**	**70.00**	**92.19**	**93.75**	**96.09**	**52.34**

(ii) OOD benchmarks

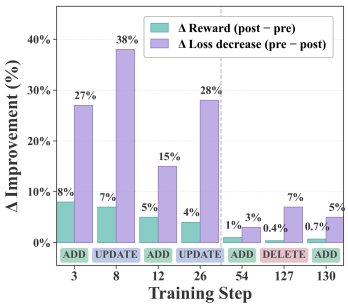
Method	MuSQ Ans EM	NQ Ans EM	MATH Acc	GPQA Acc	HEval pass@1	SciW Succ	M2W Step Acc
Qwen3.5	39.06	21.88	89.06	61.72	89.06	25.78	26.64
GRPO	73.44	79.69	87.50	79.69	89.06	39.10	33.59
Tree-GRPO	77.34	80.47	92.19	80.47	92.19	42.20	36.00
HCAPO	79.69	81.25	93.75	84.38	93.75	43.75	39.10
SkillFlow (Ours)	**85.16**	**82.81**	**96.09**	**89.84**	**98.44**	**57.81**	**51.56**

(a) Algorithm comparison on IID and OOD benchmarks (RQ5)

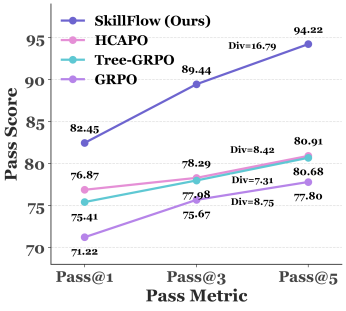
Method	collect	s/turn	s/traj
GRPO	216.8	1.79	35.6
HCAPO	137.6	1.72	48.1
Tree-GRPO	162.8	1.99	45.1
Ours	**130.5**	**1.13**	**33.9**



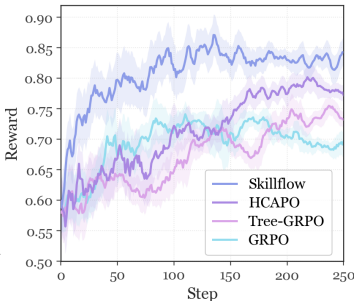
(b) Training & per-call cost



(c) Evolution events



(d) Pass@K



(e) Reward curves

Figure 5: Mechanism analysis and algorithm comparison. (a) Algorithm comparison on IID and OOD. (b) Per-call cost vs five skill-evolution baselines. (c) Skill-evolution events. (d) Pass@K vs diversity. (e) Reward curves. SkillFlow leads on accuracy, diversity, and cost simultaneously.

(Fig. 4c) is preserved across backbones, locating SkillFlow at the training-recipe level rather than backbone-specific tuning.

5.5 Component Ablation (RQ4)

Table 2 reports a leave-one-out ablation mapping onto claims C1–C3. *Loss (C1):* replacing TTB with GRPO hurts most on diversity-sensitive tasks (AIME, WebShop, ALFWorld) — mode collapse, not reward-curve shape, is the dominant failure mode. *Credit (C2):* removing P_ϕ hurts harder multi-step tasks (AIME, WebShop, ScienceWorld, Mind2Web) more than fact-based QA, where per-step credit matters most for identifying decisive decisions. *Evolution (C3):* ablating any of the three flow signals *when*, *where*, or *what* independently degrades performance, indicating none is redundant.

5.6 Algorithm Comparison and Computational Cost (RQ5)

Holding the backbone and training data fixed, SkillFlow leads GRPO, Tree-GRPO [Ji et al., 2025], and HCAPO [Tan et al., 2026] on all 14 benchmarks (Figure 5(a)); Figure 5(d,e) explains the gap: REINFORCE-style methods reinforce a single mode and hit a Pass@5 ceiling at low diversity, while reward-proportional sampling keeps multiple high-reward paths alive (same ranking holds OOD). Skill-evolution events concentrate at TTB plateaus (Fig. 5(c)), confirming the plateau-driven trigger of §4.3. On cost (Figure 5(b)), SkillFlow uses the fewest tokens and lowest time against five skill-evolution baselines [Jiang et al., 2026a, Yang et al., 2026b, Ma et al., 2026, Alzubi et al., 2026, Xia et al., 2026]; the ∼32% / ∼35% saving over SkillRL pins the gain on flow-signal-driven evolution.

6 Conclusion

SkillFlow unifies orchestration training and recursive skill evolution under TTB: reward-proportional sampling preserves diversity, hindsight backward gives zero-cost per-step credit, and flow diagnostics drive skill curation. Across 14 benchmarks it outperforms direct-LLM, RL, and skill-evolution baselines on accuracy, diversity, and cost.

References

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. Evoskill: Automated skill discovery for multi-agent systems. *arXiv preprint arXiv:2603.02766*, 2026.

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in neural information processing systems*, 34:27381–27394, 2021.

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

Yueyang Cang, Xiaoteng Zhang, Erlu Zhao, Zehua Ji, Yuhang Liu, Yuchen He, Zhiyuan Ning, Chen Yijun, Wenge Que, and Li Shi. Graph-grpo: Stabilizing multi-agent topology learning via group relative policy optimization. *arXiv preprint arXiv:2603.02701*, 2026.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*, 2023.

Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. Reinforcement learning for long-horizon interactive llm agents. *arXiv preprint arXiv:2502.01600*, 2025.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Pedro Dall'Antonia, Tiago da Silva, Daniel Csillag, Salem Lahlou, and Diego Mesquita. Avoid what you know: Divergent trajectory balance for gflownets. *arXiv preprint arXiv:2602.17827*, 2026.

Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, et al. Multi-agent collaboration via evolving orchestration. *arXiv preprint arXiv:2505.19591*, 2025.

Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, et al. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems. *arXiv preprint arXiv:2508.07407*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The twelfth international conference on learning representations*, 2023.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. *arXiv preprint arXiv:2408.08435*, 2024.

Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for llm agent reinforcement learning. *arXiv preprint arXiv:2509.21240*, 2025.

Guanyu Jiang, Zhaochen Su, Xiaoye Qu, and Yi R Fung. Xskill: Continual learning from experience and skills in multimodal agents. *arXiv preprint arXiv:2603.12056*, 2026a.

Yanna Jiang, Delong Li, Haiyu Deng, Baihe Ma, Xu Wang, Qin Wang, and Guangsheng Yu. Sok: Agentic skills–beyond tool use in llm agents. *arXiv preprint arXiv:2602.20867*, 2026b.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *The twelfth international conference on learning representations*, 2023.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, 2017.

Mingze Kong, Zikun Qu, Zhongquan Zhou, Pengyu Liang, Xiang Li, Zhiwei Shang, Zhi Hong, Kaiyu Huang, Zhiyong Wang, and Zhongxiang Dai. Workflow-r1: Group sub-sequence policy optimization for multi-turn workflow construction. *arXiv preprint arXiv:2602.01202*, 2026.

Long Li, Zhijian Zhou, Jiaran Hao, Jason Klein Liu, Yanting Miao, Wei Pang, Xiaoyu Tan, Wei Chu, Zhe Wang, Shirui Pan, et al. The choice of divergence: A neglected key to mitigating diversity collapse in reinforcement learning with verifiable reward. *arXiv preprint arXiv:2509.07430*, 2025a.

Xiaoxiao Li. When single-agent with skills replace multi-agent systems and when they fail. *arXiv preprint arXiv:2601.04748*, 2026.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Zou, and Pan Lu. In-the-flow agentic system optimization for effective planning and tool use. *arXiv preprint arXiv:2510.05592*, 2025b.

Shuo Liu, Zeyu Liang, Xueguang Lyu, and Christopher Amato. Llm collaboration with multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 32150–32158, 2026.

Xiaoqian Liu, Ke Wang, Yuchuan Wu, Fei Huang, Yongbin Li, Junge Zhang, and Jianbin Jiao. Agentic reinforcement learning with implicit step rewards. *arXiv preprint arXiv:2509.19199*, 2025.

Ziyu Ma, Shidong Yang, Yuxiang Ji, Xucong Wang, Yong Wang, Yiming Hu, Tongwen Huang, and Xiangxiang Chu. Skillclaw: Let skills evolve collectively with agentic evolver. *arXiv preprint arXiv:2604.08377*, 2026.

Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35: 5955–5967, 2022.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*, 2024.

Jiangweizhi Peng, Yuanxin Liu, Ruida Zhou, Charles Fleming, Zhaoran Wang, Alfredo Garcia, and Mingyi Hong. Hiper: Hierarchical reinforcement learning with explicit credit assignment for large language model agents. *arXiv preprint arXiv:2602.16165*, 2026.

Chen Qian, Zihao Xie, Yifei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, et al. Scaling large language model-based multi-agent collaboration. *arXiv preprint arXiv:2406.07155*, 2024.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The twelfth international conference on learning representations*, 2023.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Shuaijie She, Junxiao Liu, Yifeng Liu, Jiajun Chen, Xin Huang, and Shujian Huang. R-prm: Reasoning-driven process reward modeling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 13449–13462, 2025.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Jianghao Su, Xia Zeng, Luhui Liu, Chao Luo, Ye Chen, and Zhuoran Zhuang. Enhancing agentic rl with progressive reward shaping and value-based sampling policy optimization. *arXiv preprint arXiv:2512.07478*, 2025.

Jinwei Su, Qizhen Lan, Yinghui Xia, Lifan Sun, Weiyou Tian, Tianyu Shi, and Lewei He. Difficulty-aware agentic orchestration for query-specific multi-agent workflows. In *Proceedings of the ACM Web Conference 2026*, pages 2060–2070, 2026.

Hui-Ze Tan, Xiao-Wen Yang, Hao Chen, Jie-Jing Shao, Yi Wen, Yuteng Shen, Weihong Luo, Xiku Du, Lan-Zhe Guo, and Yu-Feng Li. Hindsight credit assignment for long-horizon llm agents. *arXiv preprint arXiv:2603.08754*, 2026.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Chenxi Wang, Zhuoyun Yu, Xin Xie, Wuguannan Yao, Runnan Fang, Shuofei Qiao, Kexin Cao, Guozhou Zheng, Xiang Qi, Peng Zhang, et al. Skillx: Automatically constructing skill knowledge bases for agents. *arXiv preprint arXiv:2604.04804*, 2026a.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, et al. Skill-sd: Skill-conditioned self-distillation for multi-turn llm agents. *arXiv preprint arXiv:2604.10674*, 2026b.

Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. Reinforcement learning for self-improving agent with skill library. *arXiv preprint arXiv:2512.17102*, 2025a.

Xiangwei Wang, Wei Wang, Ken Chen, Nanduni Nimalsiri, and Saman Halgamuge. Discovering process-outcome credit in multi-step llm reasoning. *arXiv preprint arXiv:2602.01034*, 2026c.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*, 2024a.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. Openhands: An open platform for ai software developers as generalist agents. *arXiv preprint arXiv:2407.16741*, 2024b.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025b.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First conference on language modeling*, 2024.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, et al. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026.

Renjun Xu and Yang Yan. Agent skills for large language models: Architecture, acquisition, security, and the path forward. *arXiv preprint arXiv:2602.12430*, 2026.

Matthew YR Yang, Hao Bai, Ian Wu, Gene Yang, Amrith Setlur, and Aviral Kumar. Int: Self-proposed interventions enable credit assignment in llm reasoning. *arXiv preprint arXiv:2601.14209*, 2026a.

Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, et al. Autoskill: Experience-driven lifelong learning via skill self-evolution. *arXiv preprint arXiv:2603.01145*, 2026b.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380, 2018.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022b.

Fangxu Yu, Lai Jiang, Haoqiang Kang, Shibo Hao, and Lianhui Qin. Flow of reasoning: Training llms for divergent reasoning with minimal examples. *arXiv preprint arXiv:2406.05673*, 2024.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, et al. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3053–3077, 2024.

David W Zhang, Corrado Rainone, Markus Peschl, and Roberto Bondesan. Robust scheduling with gflownets. *arXiv preprint arXiv:2302.05446*, 2023.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, et al. Evoskills: Self-evolving agent skills via co-evolutionary verification. *arXiv preprint arXiv:2604.01687*, 2026a.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow generation. In *The Thirteenth International Conference on Learning Representations*, 2024.

Mingda Zhang, Haoran Luo, Tiesunlong Shen, Qika Lin, Xiaoying Tang, Rui Mao, and Erik Cambria. Flowsteer: Interactive agentic workflow orchestration via end-to-end reinforcement learning. *arXiv preprint arXiv:2602.01664*, 2026b.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 10495–10516, 2025.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Gptswarm: Language agents as optimizable graphs. In *Forty-first International Conference on Machine Learning*, 2024.

Appendix: Theoretical Foundations of SkillFlow

This appendix provides a complete theoretical treatment of SkillFlow's flow-based learning framework, following the progressive development style of GFlowNet Foundations (Bengio et al., JMLR 2023). We establish the mathematical foundations for flow networks on DAGs, derive the Tempered Trajectory Balance (TTB) loss, prove convergence properties, and detail all supporting theoretical results. Each section builds explicitly on previous results, providing complete proofs with intermediate steps.

A Flow-Theoretic Foundations

This section introduces the fundamental concepts of flow networks on directed acyclic graphs (DAGs) and establishes the connection to reward-proportional sampling.

A.1 Flow Networks on DAGs

We begin with the basic definitions of flow networks.

Definition 1 (Flow Network)**.** *A flow network is a tuple $(\mathcal{G}, F)$ where $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ is a DAG with a unique source node $s_0 \in \mathcal{V}$ having no incoming edges and a set of terminal nodes $\mathcal{X} \subset \mathcal{V}$ having no outgoing edges, and $F : \mathcal{A} \to \mathbb{R}_{\geq 0}$ is a non-negative edge flow function assigning a non-negative real number to each directed edge.*

For each state $s \in \mathcal{V}$, we denote by $\mathrm{Pa}(s)$ the set of predecessors (parents) and $\mathrm{Ch}(s)$ the set of successors (children). We now introduce the crucial concept of state flow.

Definition 2 (State Flow and Flow Conservation)**.** *For any non-terminal state $s \in \mathcal{V} \setminus \mathcal{X}$, the **state flow** $F(s) \in \mathbb{R}_{\geq 0}$ is defined implicitly by the flow conservation law:*

$$F(s) \;\coloneqq\; \sum_{s' \in \mathrm{Ch}(s)} F(s \to s') \;=\; \sum_{s' \in \mathrm{Pa}(s)} F(s' \to s). \tag{14}$$

That is, the total flow entering a non-terminal state equals the total flow leaving it. This is the fundamental conservation law of flow networks, analogous to Kirchhoff's current law in electrical networks.

Remark 1. *Flow conservation is the defining property that relates edge flows to state flows. By Definition 2, we may consistently compute $F(s)$ by summing either incoming or outgoing edge flows.*

The partition function represents the total flow circulating through the network.

Definition 3 (Partition Function and Total Flow)**.** *The partition function Z is the total flow in the network, equal to the source flow:*

$$Z \;\coloneqq\; F(s_0) \;=\; \sum_{s' \in \mathrm{Ch}(s_0)} F(s_0 \to s') \;=\; \sum_{x \in \mathcal{X}} F(x). \tag{15}$$

The last equality (terminal flow equals source flow) follows from recursive application of flow conservation at each state: every unit of flow leaving the source must traverse some path and arrive at some terminal.

We now introduce policies derived from flow normalization.

Definition 4 (Forward Policy and Backward Policy)**.** *Given a flow F, we define:*

Forward policy*: $P_F(s' \mid s) = \frac{F(s \to s')}{F(s)}$ for each edge $s \to s'$ where s is non-terminal.*

Backward policy*: $P_B(s \mid s') = \frac{F(s \to s')}{F(s')}$ for each edge $s \to s'$ where $s' \in \mathcal{V} \setminus \{\text{source}\}$.*

By flow conservation (Definition 2), both P_F and P_B are valid probability distributions: $\sum_{s'} P_F(s' \mid s) = 1$ and $\sum_s P_B(s \mid s') = 1$.

We now define trajectory flow, which is essential for understanding how the forward policy generates trajectories.

Definition 5 (Trajectory and Trajectory Flow)**.** *A complete trajectory is a path $\tau = (s_0 \to s_1 \to \cdots \to s_T = x)$ from the source s_0 to some terminal $x \in \mathcal{X}$. The trajectory flow is:*

$$F(\tau) \;\coloneqq\; Z \cdot \prod_{t=1}^{T} P_F(s_t \mid s_{t-1}). \tag{16}$$

Lemma 4 (Trajectory Flow Identity)**.** *On a general DAG, the trajectory flow $F(\tau) = Z \cdot \prod_{t=1}^{T} P_F(s_t \mid s_{t-1})$ admits no further factorization in terms of edge or state flows alone; in particular, $F(\tau) \neq \prod_{t=1}^{T} F(s_{t-1} \to s_t)$ in general because edge flows and state flows are distinct quantities. When the DAG additionally satisfies that each non-source state has a unique parent (as in SkillFlow's tree-structured $\mathcal{G}$), $F(s_{t-1} \to s_t) = F(s_t)$ holds for every edge (all flow into s_t comes from the single edge $s_{t-1} \to s_t$), and the trajectory flow telescopes to the terminal state flow:*

$$F(\tau) \;=\; Z \cdot \prod_{t=1}^{T} \frac{F(s_t)}{F(s_{t-1})} \;=\; \frac{F(s_T)}{F(s_0)} \cdot Z \;=\; F(s_T), \tag{17}$$

using the standard ratio-product telescope on state *flows together with $F(s_0) = Z$ (Definition 3).*

We now establish the key decomposition of state flow.

Proposition 5 (Flow Decomposition)**.** *For any state $s \in \mathcal{V}$, the state flow equals the sum of flows of all trajectories passing through s:*

$$F(s) = \sum_{\tau \ni s} F(\tau), \tag{18}$$

where $\{\tau \ni s\}$ denotes all complete trajectories passing through state s. In particular, for terminal states:

$$F(x) = \sum_{\tau : s_T = x} F(\tau), \tag{19}$$

i.e., the flow into a terminal equals the sum of flows of trajectories terminating at that state.

Proof. We prove by structural induction on the depth of states in the DAG, where depth is the longest path length from the source.

Base case (depth$(s) = 0$): $s = s_0$ is the source. By Definition 3, $F(s_0) = Z = \sum_\tau F(\tau)$, since every trajectory starts at s_0 and summing over all complete trajectories gives the total flow.

Inductive step: Assume the claim holds for all states at depth $< d$. Consider a state s at depth d. By Definition 2 (flow conservation):

$$F(s) = \sum_{s' \in \mathrm{Pa}(s)} F(s' \to s). \tag{20}$$

By Definition 4, $F(s' \to s) = F(s') \cdot P_F(s \mid s')$. By the inductive hypothesis, $F(s') = \sum_{\tau' \ni s'} F(\tau')$, so:

$$F(s) = \sum_{s' \in \mathrm{Pa}(s)} F(s') \cdot P_F(s \mid s') \tag{21}$$

$$= \sum_{s' \in \mathrm{Pa}(s)} \left(\sum_{\tau' \ni s'} F(\tau') \right) \cdot P_F(s \mid s'). \tag{22}$$

Each trajectory τ' through s' can be extended to a unique trajectory through s by following the edge $s' \to s$. Since every complete trajectory through s passes through exactly one parent $s' \in \mathrm{Pa}(s)$, the above sum equals:

$$F(s) = \sum_{\tau \ni s} F(\tau). \tag{23}$$

□

A.2 Reward-Matching and GFlowNet Sampling

We now establish when a flow induces sampling proportional to a reward function.

Definition 6 (Reward-Matching Flow)**.** *Given a strictly positive reward function $R : \mathcal{X} \to \mathbb{R}_{>0}$ on terminal states, a flow F is **reward-matching** if:*

$$F(x) = R(x) \quad \textit{for all } x \in \mathcal{X}. \tag{24}$$

That is, the flow into each terminal state equals the reward of that terminal.

The central theorem of GFlowNets states that reward-matching flows induce a distribution over terminals proportional to rewards.

Theorem 6 (GFlowNet Sampling Property)**.** *If F is reward-matching with respect to $R : \mathcal{X} \to \mathbb{R}_{>0}$, then sampling a complete trajectory by following the forward policy P_F produces a distribution over terminals given by:*

$$P_F(x) \;\coloneqq\; \Pr_{\tau \sim P_F}[s_T = x] = \frac{R(x)}{Z}, \tag{25}$$

where $Z = \sum_{x' \in \mathcal{X}} R(x')$ is the partition function (total reward).

Proof. Let $\tau_x = \{x\}$ denote any complete trajectory ending at x. The probability of sampling a trajectory ending at x is:

$$P_F(x) = \sum_{\tau : s_T = x} \prod_{t=1}^{|\tau|} P_F(s_t \mid s_{t-1}). \tag{26}$$

Substituting Equation (16):

$$P_F(x) = \sum_{\tau : s_T = x} \frac{F(\tau)}{Z}. \tag{27}$$

By Proposition 5, $F(x) = \sum_{\tau : s_T = x} F(\tau)$, so:

$$P_F(x) = \frac{F(x)}{Z}. \tag{28}$$

Since F is reward-matching, $F(x) = R(x)$, thus:

$$P_F(x) = \frac{R(x)}{Z} = \frac{R(x)}{\sum_{x'} R(x')}. \tag{29}$$

□

Remark 2 (Positive Reward Requirement)**.** *The requirement $R(x) > 0$ for all x is essential to Theorem 6. If $R(x) = 0$ for some terminal, the reward-matching condition $F(x) = 0$ implies no flow reaches x under any forward policy that respects Definition 4 (since $P_F(s' \mid s)$ must be derived from non-negative edge flows). In SkillFlow, trajectories with zero or negative reward must be handled via smoothing (Appendix H).*

A.3 Application to Task Orchestration

We now instantiate the abstract flow network framework to SkillFlow's task orchestration setting.

Proposition 7 (Flow Network Instantiation for Task Orchestration)**.** *The orchestration trajectory distribution is a flow network $(\mathcal{G}, F)$ where:*

- ***Vertices** $\mathcal{V}$: Each vertex is an interaction history state H_t (Definition 1, main text).*
- ***Edges** $\mathcal{A}$: Each directed edge (H_{t-1}, H_t) corresponds to an orchestration action $a_t = (\alpha_t, o_t)$, where the action type $\alpha_t \in \{\textit{skill}, \textit{act}, \textit{accept}\}$ is selected by the supervisor.*
- ***Source** s_0: The initial state $H_0 = q \oplus \mathcal{S}_{\text{ret}} \oplus \omega_q$, formed by concatenating task q, retrieved skills, and orchestration guideline.*
- ***Terminals** $\mathcal{X}$: States where $\alpha_t = \textit{accept}$ or $t = T_{\max}$ (maximum trajectory length).*

- ***Forward policy***: $P_F(H_t \mid H_{t-1}) = \pi_\theta(a_t \mid H_{t-1})$, *the learned supervisor policy.*
- ***Edge flow***: $F(H_{t-1} \to H_t) = Z \cdot P_F(H_t \mid H_{t-1})$ *for edges in sampled trajectories.*
- ***Terminal flow***: $F(x) = \tilde{R}(\tau_x)^\beta$ *for terminal* x *reached by trajectory* τ_x*, where* $\tilde{R}(\tau) = R(\tau) + \varepsilon_{\min}$ *is the smoothed reward and* $\beta > 0$ *is the temperature.*

The key property is acyclicity, established by the strictly increasing state representation.

Lemma 8 (DAG Acyclicity by State Growth)**.** *Under the definition* $H_t = H_{t-1} \oplus (r_t, a_t, o_t^{exec})$*, each edge strictly increases the state size:* $|H_t| > |H_{t-1}|$*. Therefore, the orchestration graph is acyclic: no path can visit the same state twice, as that would require returning to a state with smaller or equal size.*

This acyclicity is essential for applying flow matching algorithms, which require the DAG structure.

B Trajectory Balance and TTB Derivation

This section derives the Trajectory Balance (TB) condition and the Tempered Trajectory Balance (TTB) loss, the core objective for SkillFlow training.

Convention (DAG edge = action; node = state H_t). Throughout this appendix, all flow-theoretic quantities are defined at the *action level* of the orchestration DAG: each node s_t corresponds to the interaction history H_t (Definition 1, main text), and each directed edge (s_{t-1}, s_t) corresponds to one orchestration action $a_t = (\alpha_t, o_t)$ that itself comprises K_t tokens produced autoregressively by the LLM. The *per-token normalization* introduced in Definition 9 is a within-edge device that averages over the K_t tokens of one action to produce a single, length-robust edge log-probability; it does *not* switch the DAG to a token-level granularity.

B.1 Backward Policy Definition

The backward policy is the reverse-direction distribution derived from reward-matching flows.

Definition 7 (Backward Policy)**.** *Given a reward-matching flow* F*, the backward policy is:*

$$P_B(s \mid s') = \frac{F(s \to s')}{F(s')}, \quad \textit{for } s \in \mathrm{Pa}(s'). \tag{30}$$

By flow conservation (Definition 2), $\sum_{s \in \mathrm{Pa}(s')} P_B(s \mid s') = \frac{1}{F(s')} \sum_s F(s \to s') = \frac{F(s')}{F(s')} = 1$*, so* $P_B(\cdot \mid s')$ *is a valid probability distribution over parents.*

The backward policy encodes the reverse probability of transitioning from s' back to each parent s. For learning, we will use a learned backward policy P_ϕ parameterized by ϕ.

Remark 3 (SkillFlow's Hindsight-Conditioned Backward)**.** *In SkillFlow, the learned backward policy* P_ϕ *conditions on the* hindsight state $\tilde{s}_t \coloneqq H_{t-1} \oplus o_t^{exec}$*, which augments the pre-action history with the post-action execution observation:*

$$P_\phi(a_t \mid \tilde{s}_t) \;=\; P_\phi(a_t \mid H_{t-1} \oplus o_t^{exec}). \tag{31}$$

This hindsight conditioning is well-defined as a backward policy on the action-DAG: by Eq. (4)*, the post-action state* $s_t = H_t$ *deterministically encodes* (a_t, o_t^{exec}) *given* $s_{t-1} = H_{t-1}$*, so conditioning on* s_t *is equivalent to conditioning on* $(\tilde{s}_t, a_t)$*. The information asymmetry* $\tilde{s}_t \supset s_{t-1}$ *is what makes the step-importance ratio* $I(t) = \pi_\theta(a_t \mid r_t, H_{t-1}) / P_\phi(a_t \mid \tilde{s}_t)$ *a non-trivial credit signal (Lemma 14 in B.4).*

B.2 Trajectory Balance Condition and Full Derivation

The Trajectory Balance (TB) condition is the equivalence characterizing when a flow is reward-matching.

Theorem 9 (Trajectory Balance (Malkin et al., 2022))**.** *A flow F on a DAG is reward-matching iff for every complete trajectory $\tau = (s_0, s_1, \ldots, s_T)$ with $s_T \in \mathcal{X}$:*

$$Z \cdot \prod_{t=1}^{T} P_F(s_t \mid s_{t-1}) = F(s_T) \cdot \prod_{t=1}^{T} P_B(s_{t-1} \mid s_t). \tag{32}$$

Equivalently, in log-space:

$$\log Z + \sum_{t=1}^{T} \log P_F(s_t \mid s_{t-1}) = \log F(s_T) + \sum_{t=1}^{T} \log P_B(s_{t-1} \mid s_t). \tag{33}$$

Proof. We prove both directions explicitly.

Forward direction (F reward-matching $\Rightarrow$ TB holds):

Assume F is reward-matching, so $F(x) = R(x)$ for all terminals x. Expand the LHS and RHS of Equation (32):

$$\text{LHS} = Z \cdot \prod_{t=1}^{T} \frac{F(s_{t-1} \to s_t)}{F(s_{t-1})}, \tag{34}$$

$$\text{RHS} = R(s_T) \cdot \prod_{t=1}^{T} \frac{F(s_{t-1} \to s_t)}{F(s_t)}. \tag{35}$$

Taking the ratio LHS/RHS:

$$\frac{\text{LHS}}{\text{RHS}} = \frac{Z}{R(s_T)} \cdot \prod_{t=1}^{T} \frac{F(s_t)}{F(s_{t-1})}. \tag{36}$$

The product telescopes:

$$\prod_{t=1}^{T} \frac{F(s_t)}{F(s_{t-1})} = \frac{F(s_1)}{F(s_0)} \cdot \frac{F(s_2)}{F(s_1)} \cdots \frac{F(s_T)}{F(s_{T-1})} = \frac{F(s_T)}{F(s_0)} = \frac{R(s_T)}{Z}, \tag{37}$$

where we used $F(s_0) = Z$ (Definition 3) and $F(s_T) = R(s_T)$ (reward-matching). Thus $\text{LHS}/\text{RHS} = \frac{Z}{R(s_T)} \cdot \frac{R(s_T)}{Z} = 1$, so LHS = RHS.

Reverse direction (TB holds $\Rightarrow$ F reward-matching):

Assume TB holds for every trajectory. Sum Equation (32) over all trajectories ending at a fixed terminal x:

$$Z \cdot \sum_{\tau : s_T = x} \prod_{t=1}^{|\tau|} P_F(s_t \mid s_{t-1}) = \sum_{\tau : s_T = x} F(x) \cdot \prod_{t=1}^{|\tau|} P_B(s_{t-1} \mid s_t). \tag{38}$$

The LHS is $Z \cdot P_F(x)$ by definition of the forward policy probability. For the RHS, by the product structure and properties of the backward policy (which forms a distribution over reverse trajectories), the sum $\sum_{\tau : s_T = x} \prod_t P_B(s_{t-1} \mid s_t) = 1$ (the reverse-direction probability of reaching x from all trajectories sums to 1). Thus:

$$Z \cdot P_F(x) = F(x). \tag{39}$$

By Theorem 6, this implies $P_F(x) = F(x)/Z$, which combined with $Z = \sum_{x'} F(x')$ gives $F(x) = R(x)$. □

B.3 TTB Loss Derivation with Temperature and Length Normalization

We now derive the Tempered Trajectory Balance loss as the squared, length-normalized log-space TB residual on the action-DAG, with two technical devices: (i) reasoning r_t enters the forward conditional but is treated as fixed context (Lemma 13, B.4); (ii) each action edge's log-probability is averaged across its K_t tokens (per-token tempering, Lemma 15, B.4). Both devices are formally justified in B.4 and do *not* alter the action-level DAG structure.

Definition 8 (Per-Token-Tempered Edge Log-Probabilities)**.** *Let action a_t comprise K_t tokens $\mathrm{tok}_1^{(t)}, \ldots, \mathrm{tok}_{K_t}^{(t)}$ produced autoregressively. The per-token-tempered forward and backward edge log-probabilities are:*

$$\widetilde{\log \pi}_\theta(a_t \mid r_t, H_{t-1}) \coloneqq \frac{1}{K_t} \sum_{j=1}^{K_t} \log \pi_\theta\big(\mathrm{tok}_j^{(t)} \,\big|\, \mathrm{tok}_{<j}^{(t)},\, r_t,\, H_{t-1}\big), \tag{40}$$

$$\widetilde{\log P}_\phi(a_t \mid H_{t-1} \oplus o_t^{exec}) \coloneqq \frac{1}{K_t} \sum_{j=1}^{K_t} \log P_\phi\big(\mathrm{tok}_j^{(t)} \,\big|\, \mathrm{tok}_{<j}^{(t)},\, H_{t-1} \oplus o_t^{exec}\big). \tag{41}$$

Each is the geometric mean (in log-space) of token probabilities along the K_t tokens of one action edge; the edge thus carries a single, length-robust log-probability.

Definition 9 (TTB Residual)**.** *Given the per-token-tempered edge log-probabilities of Definition 8, the trajectory balance residual is:*

$$\Delta(\tau) \;\coloneqq\; \log Z_\theta(q) + \sum_{t=1}^{T} \widetilde{\log \pi}_\theta(a_t \mid r_t, H_{t-1}) - \beta \log \tilde{R}(\tau) - \sum_{t=1}^{T} \widetilde{\log P}_\phi(a_t \mid H_{t-1} \oplus o_t^{exec}), \tag{42}$$

where:

- *$Z_\theta(q)$ is a task-conditioned partition function (learned parameter),*
- *$\beta > 0$ is the temperature parameter controlling diversity,*
- *$\tilde{R}(\tau) = R(\tau) + \varepsilon_{\min}$ is the smoothed reward,*
- *r_t is the reasoning emitted at step t, treated as fixed conditioning context (Lemma 13),*
- *$T = |\tau|$ is the trajectory length, i.e., the number of action edges.*

At the optimum $\Delta(\tau) = 0$, the induced tempered *forward distribution satisfies $\tilde{\pi}_\theta(\tau) \propto \tilde{R}(\tau)^\beta$ (Lemma 15).*

The TB condition in log-space (Equation (33)) requires $\Delta(\tau) = 0$. To enforce this across all trajectories, we use a regression loss:

Definition 10 (Tempered Trajectory Balance Loss)**.** *The TTB loss is:*

$$\mathcal{L}_{\mathrm{TTB}}(\tau) = \left(\frac{\Delta(\tau)}{T}\right)^2, \tag{43}$$

with $\Delta(\tau)$ as in Definition 9 and division by T comparing trajectories of different lengths on equal footing.

Lemma 10 (Length Normalization Property)**.** *Each tempered edge log-probability is $O(1)$ by per-token averaging (Definition 8), so $|\sum_{t=1}^{T} \widetilde{\log \pi}_\theta(a_t \mid r_t, H_{t-1})| = O(T)$ and unmormalized $|\Delta(\tau)| \propto T$ for typical trajectories. Dividing by T before squaring penalizes the* average *per-step balance rather than the cumulative balance, keeping $|\Delta(\tau)/T| = O(1)$ and stabilizing gradients across length variations.*

Proposition 11 (TTB Self-Annealing Property)**.** *Treating $T = |\tau|$ as fixed for a given τ, the gradient of the TTB loss satisfies:*

$$\nabla_\theta \mathcal{L}_{\mathrm{TTB}}(\tau) = 2 \cdot \frac{\Delta(\tau)}{T} \cdot \nabla_\theta \left(\frac{\Delta(\tau)}{T}\right) = \underbrace{\frac{2\,\Delta(\tau)}{T^2}}_{\textit{length-scaled annealing factor}} \cdot \nabla_\theta \Delta(\tau). \tag{44}$$

As $\Delta(\tau) \to 0$, the prefine shrinks and $\|\nabla_\theta \mathcal{L}_{\mathrm{TTB}}\| \to 0$ automatically. Because T varies across trajectories, the $1/T^2$ factor is a per-trajectory length normalizer and cannot be absorbed into a global constant.

The temperature β is a critical hyperparameter controlling the diversity-quality tradeoff:

Lemma 12 (Temperature Effect on Policy)**.** *At convergence,* $\pi^*(\tau) \propto \tilde{R}(\tau)^\beta$.

- *As* $\beta \to 0^+$*: the policy approaches uniform distribution over trajectories (maximum diversity, ignoring rewards).*
- *As* $\beta \to \infty$*: the policy concentrates on the single highest-reward trajectory (maximum quality, no diversity).*
- *Intermediate* β *trades off diversity and quality.*

B.4 Conditional TB, Hindsight Backward, and Per-Token Tempering

The TTB residual in Definition 9 departs from textbook TB in three ways: (i) the forward policy conditions on reasoning r_t; (ii) the backward policy conditions on the hindsight state $H_{t-1} \oplus o_t^{\text{exec}}$; (iii) each edge log-probability is per-token averaged. We establish three lemmas showing that none of these alters the action-DAG structure or the reward-matching guarantee.

(i) Reasoning as fixed context.

Lemma 13 (TB on the Action Sub-DAG with Reasoning as Fixed Context)**.** *The hierarchical step policy (Eq. 5, main text) factorizes as*

$$\pi_\theta(r_t, a_t \mid H_{t-1}) \;=\; \pi_\theta(r_t \mid H_{t-1}) \cdot \pi_\theta(a_t \mid r_t, H_{t-1}). \tag{45}$$

Conditioning on the realized reasoning sequence $\{r_t\}_{t=1}^T$ *and execution observations* $\{o_t^{exec}\}_{t=1}^T$*, only the action choices* $\{a_t\}$ *remain random. The conditional trajectory probability is*

$$P_\theta\big(\tau \,\big|\, \{r_t\}, \{o_t^{exec}\}\big) \;=\; \prod_{t=1}^{T} \pi_\theta(a_t \mid r_t, H_{t-1}), \tag{46}$$

which is precisely the forward policy on the action sub-DAG $\mathcal{G}^{\text{act}} \subset \mathcal{G}$*. The TB condition (Theorem 9) is a per-trajectory equality and therefore applies pointwise to each realized* $(\{r_t\}, \{o_t^{exec}\})$ *context, yielding the log-space form*

$$\log Z_\theta(q) + \sum_{t=1}^{T} \log \pi_\theta(a_t \mid r_t, H_{t-1}) \;=\; \log F(\tau) + \sum_{t=1}^{T} \log P_B(s_{t-1} \mid s_t), \tag{47}$$

which matches Eq. 8 with r_t *in the conditional. Reward-matching of the resulting flow on* $\mathcal{G}^{\text{act}}$ *is therefore equivalent to TB holding on every trajectory at every realized reasoning context.*

Proof. Direct from the hierarchical factorization and Theorem 9. Reasoning r_t enters $\pi_\theta(a_t \mid r_t, H_{t-1})$ as a determining context; once realized, it is treated as part of the conditioning state for the action edge, exactly as the source-state $s_{t-1} = H_{t-1}$ is. The sub-DAG $\mathcal{G}^{\text{act}}$ inherits its DAG structure from $\mathcal{G}$ (Theorem 26). □

(ii) Hindsight-asymmetric backward policy.

Lemma 14 (Hindsight-Conditioned Backward Equals Standard Backward on the Augmented State)**.** *Define the hindsight-enriched pre-action state* $\tilde{s}_t := H_{t-1} \oplus o_t^{exec}$*. By Eq. 4, the post-action state* $s_t = H_t$ *deterministically encodes the joint* $(H_{t-1}, a_t, o_t^{exec})$ *given* $s_{t-1} = H_{t-1}$*. Therefore conditioning on* s_t *is equivalent to conditioning on* $(\tilde{s}_t, a_t)$*, and*

$$P_B(s_{t-1} \mid s_t) \;=\; P_\phi(a_t \mid \tilde{s}_t) \;=\; P_\phi(a_t \mid H_{t-1} \oplus o_t^{exec}), \tag{48}$$

in the sense that both express the same conditional reverse mass. Substituting this equality into Eq. (47) *yields the SkillFlow TB condition*

$$\log Z_\theta(q) + \sum_{t=1}^{T} \log \pi_\theta(a_t \mid r_t, H_{t-1}) \;=\; \beta \log \tilde{R}(\tau) + \sum_{t=1}^{T} \log P_\phi(a_t \mid H_{t-1} \oplus o_t^{exec}), \tag{49}$$

where $\log F(\tau) = \beta \log \tilde{R}(\tau)$ *at the reward-matching terminal. The information asymmetry* $\tilde{s}_t \supset s_{t-1}$ *(post-execution observation* o_t^{exec} *added) is precisely what gives the step-importance ratio* $I(t) = \pi_\theta(a_t \mid r_t, H_{t-1}) / P_\phi(a_t \mid \tilde{s}_t)$ *a non-trivial credit signal: high* $I(t)$ *marks decisions whose quality became clear only after execution.*

Proof. By the strict-history-growth lemma (Lemma 8), H_t uniquely determines its parent H_{t-1} and the appended triple $(r_t, a_t, o_t^{\text{exec}})$. Hence $s_t \mapsto (s_{t-1}, r_t, a_t, o_t^{\text{exec}})$ is a bijection. Marginalizing over r_t (which is fixed in the conditional TB of Lemma 13) leaves the joint $(s_{t-1}, a_t, o_t^{\text{exec}})$, equivalent to $(\tilde{s}_t, a_t)$. The reverse conditional therefore equals $P_\phi(a_t \mid \tilde{s}_t)$ by definition of P_ϕ as a learned discriminative reverse model on the hindsight state. □

(iii) Per-token tempering preserves convergence semantics.

Lemma 15 (Per-Token Tempering as Edge-Level Geometric-Mean Rescaling)**.** *Let* $\widetilde{\log \pi}_\theta(a_t \mid r_t, H_{t-1})$ *and* $\widetilde{\log P}_\phi(a_t \mid H_{t-1} \oplus o_t^{exec})$ *be as in Definition 8, and define the tempered edge probabilities*

$$\tilde{\pi}_\theta(a_t \mid r_t, H_{t-1}) \coloneqq \exp\big(\widetilde{\log \pi}_\theta(a_t \mid r_t, H_{t-1})\big), \quad \tilde{P}_\phi(a_t \mid \tilde{s}_t) \coloneqq \exp\big(\widetilde{\log P}_\phi(a_t \mid \tilde{s}_t)\big). \tag{50}$$

Then:

(i) $\tilde{\pi}_\theta$ is the geometric mean of token probabilities: $\tilde{\pi}_\theta(a_t) = \prod_j \pi_\theta(\text{tok}_j)^{1/K_t}$, a length-normalized policy on the action edge.

(ii) Substituting $\tilde{\pi}_\theta, \tilde{P}_\phi$ into the TB condition (Eq. (49)) and setting $\Delta(\tau) = 0$ defines a tempered flow $\tilde{F}$ with $\tilde{F}(x) = \tilde{R}(\tau_x)^\beta$. By Theorem 6 applied to $\tilde{F}$, the induced sampling distribution satisfies $\tilde{\pi}_\theta(\tau \mid q) \propto \tilde{R}(\tau)^\beta$.

(iii) Geometric-mean tempering is a strictly monotone positive rescaling of edge probabilities; trajectory probabilities are products of edge probabilities, so the trajectory-level reward-proportional ranking is preserved: $\tilde{\pi}_\theta(\tau_1) > \tilde{\pi}_\theta(\tau_2) \Leftrightarrow \tilde{R}(\tau_1)^\beta > \tilde{R}(\tau_2)^\beta \Leftrightarrow R(\tau_1) > R(\tau_2)$ (Proposition 28).

Proof. (i) Direct from $\exp\big(\frac{1}{K_t}\sum_j \log \pi_\theta(\text{tok}_j)\big) = \prod_j \pi_\theta(\text{tok}_j)^{1/K_t}$. (ii) The TTB residual is the same algebraic identity as TB with P_F, P_B replaced by $\tilde{\pi}_\theta, \tilde{P}_\phi$; the proof of Theorem 9 carries through verbatim with these tempered policies, defining a reward-matching tempered flow. (iii) Monotone rescaling preserves $\arg\max$ and ordering. □

Remark 4 (Why Per-Token Tempering, Not Token-Level GFlowNet)**.** *The per-token sums in Eq.* (40)–(41) *average* within *a single action edge to produce one length-robust edge log-probability. The DAG nodes remain action-level states H_t; tokens are the LLM's internal autoregressive representation of one edge, not separate DAG nodes. This distinguishes our setting from token-level GFlowNet variants where each token is its own DAG edge.*

Together, Lemmas 13, 14, and 15 provide the formal justification for the SkillFlow TTB residual (Definition 9, equivalently main-text Eq. 9): it is the standard log-space TB residual on the action-DAG, evaluated with r_t-conditioned forward policy, hindsight-conditioned backward policy, and per-token-tempered edge log-probabilities.

C Entropy-Regularized RL Equivalence

This section proves the fundamental equivalence between GFlowNet training and entropy-regularized reinforcement learning.

Theorem 16 (GFlowNet-RL Equivalence)**.** *The optimal policy induced by GFlowNet training with temperature β is identical to the optimal policy of entropy-regularized maximum expected reward, with temperature parameter $T = 1/\beta$:*

$$\pi^*_{\text{GFN}}(\tau \mid q) = \frac{\tilde{R}(\tau)^\beta}{\sum_{\tau'} \tilde{R}(\tau')^\beta} = \frac{\exp\big[\beta \log \tilde{R}(\tau)\big]}{Z_\beta(q)}. \tag{51}$$

This matches the optimal policy from the entropy-regularized RL objective:

$$\pi^*_{\text{RL}}(\tau \mid q) = \arg\max_\pi \mathbb{E}_{\tau\sim\pi}\left[\log R(\tau)\right] + \frac{1}{\beta}\mathcal{H}[\pi], \tag{52}$$

where $\mathcal{H}[\pi] = -\mathbb{E}_\tau[\log \pi(\tau)]$ is the policy entropy.

Proof. The entropy-regularized RL objective (with temperature $T = 1/\beta$) is:

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\log R(\tau)\right] + T \cdot \mathcal{H}[\pi] = \mathbb{E}_{\tau\sim\pi}\left[\log R(\tau) - \log \pi(\tau)\right] + T\log(|\mathcal{T}|), \tag{53}$$

where $|\mathcal{T}|$ is the number of trajectories (constant in π).

Taking the functional derivative with respect to $\pi(\tau)$ and setting it to zero:

$$\frac{\delta J}{\delta \pi(\tau)} = \log R(\tau) - \log \pi^*(\tau) - 1 + T\log(|\mathcal{T}|) = 0. \tag{54}$$

Solving for $\pi^*(\tau)$:

$$\log \pi^*(\tau) = \log R(\tau) + T\log(|\mathcal{T}|) - 1 = \frac{1}{T}\log R(\tau) + \text{const.} \tag{55}$$

Exponentiating:

$$\pi^*(\tau) \propto \exp\left(\frac{1}{T}\log R(\tau)\right) = R(\tau)^{1/T} = R(\tau)^{\beta}. \tag{56}$$

Normalizing:

$$\pi^*(\tau) = \frac{R(\tau)^{\beta}}{\sum_{\tau'} R(\tau')^{\beta}}. \tag{57}$$

With smoothing $\tilde{R}(\tau) = R(\tau) + \varepsilon_{\min}$, this becomes:

$$\pi^*(\tau) = \frac{\tilde{R}(\tau)^{\beta}}{\sum_{\tau'} \tilde{R}(\tau')^{\beta}} = \frac{\exp[\beta \log \tilde{R}(\tau)]}{Z_{\beta}(q)}, \tag{58}$$

which exactly matches the GFlowNet sampling distribution from Theorem 6. □

Remark 5 (Practical Implications)**.** *This equivalence provides a principled interpretation of GFlowNet training: maximizing reward while maintaining policy entropy, under the weighting controlled by β. Unlike standard policy gradient methods (REINFORCE, GRPO), which optimize toward a single reward-maximum strategy, the entropy regularization preserves diverse solutions with comparable rewards.*

D Detailed Balance and Flow Metrics

This section establishes the Detailed Balance condition and derives the flow-based credit assignment metrics.

D.1 Detailed Balance Theorem

Theorem 17 (Detailed Balance)**.** *For a reward-matching flow F satisfying the Trajectory Balance condition (Theorem 9), the Detailed Balance (DB) condition holds at every edge:*

$$F(s) \cdot P_F(s' \mid s) = F(s') \cdot P_B(s \mid s'), \tag{59}$$

where $P_F(s' \mid s) = F(s \to s')/F(s)$ and $P_B(s \mid s') = F(s \to s')/F(s')$.

Proof. By definition of P_F and P_B from Definition 4:

$$\text{LHS} = F(s) \cdot \frac{F(s \to s')}{F(s)} = F(s \to s'), \tag{60}$$

$$\text{RHS} = F(s') \cdot \frac{F(s \to s')}{F(s')} = F(s \to s'). \tag{61}$$

Both sides equal the edge flow $F(s \to s')$. □

Detailed Balance is a pointwise (per-edge) condition, stronger than Trajectory Balance (which is trajectory-wise). For a reward-matching flow, DB emerges as a consequence. In SkillFlow's tree-structured DAG (each H_t has a unique parent by strict history growth), every trajectory through a state is unique, so TB at convergence uniquely determines the edge flows and DB follows. The next lemma makes this uniqueness explicit.

Lemma 18 (Tree-DAG Specialization: Uniqueness of the Reward-Matching Flow and Per-Edge DB)**.** *On a tree-structured DAG (every non-source node has a unique parent), as is the case for SkillFlow's orchestration graph by Lemma 8, a reward-matching flow F with terminal values $\{F(x) = R(x)\}_{x\in\mathcal{X}}$ is uniquely determined: for every non-terminal state s,*

$$F(s) \;=\; \sum_{x\in \mathrm{Desc}(s)\,\cap\,\mathcal{X}} R(x), \tag{62}$$

where $\mathrm{Desc}(s)$ is the set of descendants of s. Consequently, TB convergence (Theorem 9) on a tree-DAG simultaneously enforces (i) reward-matching at terminals, (ii) flow uniqueness at every state, and (iii) per-edge Detailed Balance (Theorem 17) at every edge.

Proof. We prove Eq. (62) by reverse induction on the depth of s (longest path from the source).

Base case (terminal $s = x \in \mathcal{X}$): $F(x) = R(x)$ by reward-matching, and $\mathrm{Desc}(x) \cap \mathcal{X} = \{x\}$, so Eq. (62) is the trivial identity $R(x) = R(x)$.

Inductive step: Suppose Eq. (62) holds for all states deeper than s. By flow conservation (Definition 2),

$$F(s) = \sum_{s'\in \mathrm{Ch}(s)} F(s \to s') = \sum_{s'\in \mathrm{Ch}(s)} F(s'), \tag{63}$$

where the second equality uses tree structure: each child s' has s as its unique parent, so all flow into s' comes from the single edge $s \to s'$, i.e., $F(s') = F(s \to s')$. By the inductive hypothesis,

$$F(s) = \sum_{s'\in \mathrm{Ch}(s)} \;\sum_{x\in \mathrm{Desc}(s')\,\cap\,\mathcal{X}} R(x) = \sum_{x\in \mathrm{Desc}(s)\,\cap\,\mathcal{X}} R(x), \tag{64}$$

where the last equality uses the disjoint partition $\mathrm{Desc}(s) \setminus \{s\} = \bigsqcup_{s'\in \mathrm{Ch}(s)} (\{s'\} \cup \mathrm{Desc}(s'))$, valid because subtrees of distinct children of s are disjoint on a tree.

For (iii), Theorem 17 (proved for general DAGs) yields DB at every edge once F is reward-matching. □

D.2 Flow Ratio and Step Importance

Corollary 19 (Flow Ratio)**.** *From Detailed Balance (Theorem 17), $F(s_{t-1}) \cdot P_F(s_t \mid s_{t-1}) = F(s_t) \cdot P_B(s_{t-1} \mid s_t)$. Dividing both sides by $F(s_{t-1}) \cdot P_B(s_{t-1} \mid s_t)$:*

$$\frac{F(s_t)}{F(s_{t-1})} = \frac{P_F(s_t \mid s_{t-1})}{P_B(s_{t-1} \mid s_t)}. \tag{65}$$

Substituting the SkillFlow policy realizations from Lemmas 13 and 14:

$$\frac{F(s_t)}{F(s_{t-1})} \;=\; \frac{\pi_\theta(a_t \mid r_t, H_{t-1})}{P_\phi(a_t \mid H_{t-1} \oplus o_t^{exec})}. \tag{66}$$

This leads to the key quantity for credit assignment:

Definition 11 (Step Importance)**.** *The step importance at time t is:*

$$I(t) \;\coloneqq\; \frac{F(s_t)}{F(s_{t-1})} \;=\; \frac{\pi_\theta(a_t \mid r_t, H_{t-1})}{P_\phi(a_t \mid H_{t-1} \oplus o_t^{exec})}. \tag{67}$$

This ratio quantifies the "flow amplification" at step t: how much the state flow increases (if $I(t) > 1$) or decreases (if $I(t) < 1$) due to action a_t, measured against the hindsight-asymmetric backward (Lemma 14).

Intuitively, $I(t)$ measures the information value of decision a_t:

- $I(t) \gg 1$: The decision was high-probability for the forward policy but would have been low-probability in hindsight; this decision had high impact.
- $I(t) \approx 1$: The forward and backward policies agree; the decision had typical impact.
- $I(t) \ll 1$: The decision was low-probability in hindsight relative to the forward policy; this decision was sub-optimal in retrospect.

D.3 Skill Marginal Flow

Definition 12 (Skill Marginal Flow)**.** *For a skill $s \in \mathcal{S}$, the marginal flow is the average post-action* state flow *at nodes where s is invoked, aggregated over trajectories in the batch:*

$$\hat{F}(s) \;\coloneqq\; \frac{1}{|\mathcal{B}_s|} \sum_{\tau \in \mathcal{B}_s} \sum_{t:\, a_t \text{ invokes } s} F(s_t), \tag{68}$$

where $\mathcal{B}_s \subseteq \mathcal{B}$ is the subset of sampled trajectories that invoke skill s, and the post-action state flow is recovered by telescoping the step importance from the source $F(s_0) = Z_\theta(q)$:

$$F(s_t) \;=\; Z_\theta(q) \cdot \prod_{t'=1}^{t} I(t') \quad\Longleftrightarrow\quad \log F(s_t) \;=\; \log Z_\theta(q) + \sum_{t'=1}^{t} \log I(t'). \tag{69}$$

This matches main-text Eq. 11 exactly.

Skill marginal flow is the key signal for detecting which skills contribute most to reward-proportional sampling. High $\hat{F}(s)$ indicates that s is invoked at high-flow states—which by Theorem 6 are visited with probability proportional to downstream reward. Low $\hat{F}(s)$ indicates that s either (i) is rarely invoked, (ii) is invoked along low-flow / low-reward trajectories, or (iii) is invoked at decision points where the cumulative information advantage of the policy is small (small $\sum_{t' \le t} \log I(t')$).

D.4 Zero-Cost Computation

Proposition 20 (Zero-Cost Flow Metrics)**.** *All flow metrics—state flows $F(s_t)$, step importances $I(t)$, and skill marginal flows $\hat{F}(s)$—are computable from the (per-token-tempered) forward and backward log-probabilities already evaluated during the TTB loss computation (Eq.* (43)*).*

Concretely:

$$\log I(t) = \widetilde{\log \pi}_\theta(a_t \mid r_t, H_{t-1}) - \widetilde{\log P}_\phi(a_t \mid H_{t-1} \oplus o_t^{exec}), \tag{70}$$

$$\log F(s_t) = \log Z_\theta(q) + \sum_{t'=1}^{t} \log I(t'), \tag{71}$$

$$\log \hat{F}(s) = \log Z_\theta(q) + \log\left(\frac{1}{|\mathcal{B}_s|} \sum_{\tau \in \mathcal{B}_s} \sum_{t:\, a_t \text{ invokes } s} \exp\Big(\sum_{t'=1}^{t} \log I(t') \Big) \right). \tag{72}$$

Equation (72) *is the log of an arithmetic mean (matching the linear-scale definition in Eq.* (68)*),* not *the mean-of-log: by the telescoping identity each visit contributes $F(s_t) = Z_\theta(q) \cdot \exp(\sum_{t' \le t} \log I(t'))$, and we average those visit-flows in linear space before re-taking the log. This expression is precisely the per-skill CGF at $\lambda = 1$ shifted by $\log Z_\theta(q)$ (cf. main-text Eq. 13 and the equality $\Lambda_1^{(s)} = \log \hat{F}(s) - \log Z_\theta(q)$). No additional forward or backward passes through the policy models are required.*

This zero-cost property is crucial for SkillFlow: the flow signals that drive skill evolution incur no computational overhead beyond the standard TTB loss evaluation.

D.5 Cumulant Generating Function (CGF) Properties

Section 4.3 of the main text defines, for each skill $s \in \mathcal{S}$, the per-skill cumulant generating function (CGF) of the telescoped log step-importance:

$$\Lambda_\lambda^{(s)} \;\coloneqq\; \log\left(\frac{1}{|\mathcal{B}_s|} \sum_{\tau \in \mathcal{B}_s} \sum_{t:\, a_t \text{ invokes } s} \exp\Big(\lambda \sum_{t'=1}^{t} \log I(t') \Big) \right), \qquad \lambda \in \mathbb{R}. \tag{73}$$

Let $V_s \coloneqq \{(\tau, t) \,:\, \tau \in \mathcal{B}_s,\, a_t \text{ invokes } s\}$ denote the set of *s-visits* in the batch, and for each visit $v = (\tau, t) \in V_s$ define the telescoped log-flow share

$$X_v \;\coloneqq\; \sum_{t'=1}^{t} \log I(t') \;\overset{\text{Eq. (69)}}{=}\; \log F(s_t) \;-\; \log Z_\theta(q). \tag{74}$$

This appendix proves the four properties of $\Lambda_\lambda^{(s)}$ used by the curation operator Φ (Eq. 13, main text): convexity, $G(s)$ as the visit-mean of X, $\Lambda_1^{(s)}$ as the centered log skill marginal flow, and the Jensen-gap cumulant expansion.

Atomicity assumption. By the *atomic-tip* property in main-text §4.3, each tip s is self-contained and independently composable; we assume each atomic tip is invoked *at most once* per trajectory in $\mathcal{B}_s$. Under this assumption, $|V_s| = |\mathcal{B}_s|$ and Eq. (73) reduces to the standard sample CGF over the empirical visit-distribution:

$$\Lambda_\lambda^{(s)} \;=\; \log\!\left(\frac{1}{|V_s|}\sum_{v\in V_s} e^{\lambda X_v}\right) \;=\; \log\,\mathbb{E}_{V_s}\!\big[e^{\lambda X}\big]. \tag{75}$$

We treat $\Lambda_\lambda^{(s)}$ as the standard CGF over $\{X_v\}_{v\in V_s}$ throughout.

Lemma 21 (Convexity of the CGF in λ)**.** *$\Lambda_\lambda^{(s)}$ is convex in $\lambda \in \mathbb{R}$.*

Proof. By Eq. (75), $\Lambda_\lambda^{(s)} = \log\sum_v e^{\lambda X_v} - \log|V_s|$. Each λX_v is affine in λ, $\log\sum e^{(\cdot)}$ (log-sum-exp) is a standard convex function of its arguments, and convexity is preserved under affine pre-composition and constant shift. Hence $\Lambda_\lambda^{(s)}$ is convex in λ. □

Lemma 22 (Mean Log-Flow Identity for $G(s)$)**.**

$$G(s) \;\coloneqq\; \frac{\partial\Lambda_\lambda^{(s)}}{\partial\lambda}\bigg|_{\lambda=0} \;=\; \frac{1}{|V_s|}\sum_{v\in V_s} X_v \;=\; \mathbb{E}_{V_s}\big[\log F(s_t)\big] \;-\; \log Z_\theta(q). \tag{76}$$

That is, $G(s)$ is the visit-average of the centered log state-flow $\log F(s_t) - \log Z_\theta(q)$ over occurrences of s.

Proof. Differentiate Eq. (75) with respect to λ:

$$\frac{\partial}{\partial\lambda}\log\!\Big(\tfrac{1}{|V_s|}\sum_v e^{\lambda X_v}\Big) \;=\; \frac{\sum_v X_v\, e^{\lambda X_v}}{\sum_v e^{\lambda X_v}}. \tag{77}$$

At $\lambda = 0$ this evaluates to $\frac{\sum_v X_v}{\sum_v 1} = \frac{1}{|V_s|}\sum_v X_v$, the empirical visit-mean. Substituting Eq. (74) for X_v gives the second equality. □

Lemma 23 (CGF at $\lambda = 1$ Recovers the Skill Marginal Flow)**.**

$$\Lambda_1^{(s)} \;=\; \log\hat{F}(s) \;-\; \log Z_\theta(q), \tag{78}$$

where $\hat{F}(s)$ is the skill marginal flow (Definition 12). This recovers main-text Eq. 11.

Proof. Substituting $X_v = \log F(s_t^{(v)}) - \log Z_\theta(q)$ into Eq. (75) at $\lambda = 1$:

$$\begin{aligned}\Lambda_1^{(s)} &= \log\left(\frac{1}{|V_s|}\sum_v e^{X_v}\right) = \log\left(\frac{1}{|V_s|}\sum_v \frac{F(s_t^{(v)})}{Z_\theta(q)}\right)\\ &= \log\left(\frac{1}{Z_\theta(q)}\cdot\frac{1}{|V_s|}\sum_v F(s_t^{(v)})\right) \;=\; \log\hat{F}(s) - \log Z_\theta(q),\end{aligned}$$

where the last step uses Definition 12 and the atomicity identity $|V_s| = |\mathcal{B}_s|$. □

Lemma 24 (Jensen Inequality on the CGF)**.** *For every skill s with $V_s \neq \emptyset$,*

$$G(s) \;\le\; \Lambda_1^{(s)}, \tag{79}$$

with equality if and only if X_v is constant across all visits $v \in V_s$.

Proof. Apply Jensen's inequality to the convex function exp:

$$\exp\Big(\tfrac{1}{|V_s|}\sum_v X_v\Big) \;\le\; \tfrac{1}{|V_s|}\sum_v e^{X_v}, \tag{80}$$

with equality iff X_v is constant. Taking log on both sides yields $G(s) \le \log\big(\tfrac{1}{|V_s|}\sum_v e^{X_v}\big) = \Lambda_1^{(s)}$, where the equality on the right uses Eq. (75). □

Proposition 25 (Jensen Gap as Cumulant Expansion)**.** *The Jensen gap admits the cumulant expansion*

$$\Lambda_1^{(s)} - G(s) \;=\; \tfrac{1}{2}\,\mathrm{Var}_{V_s}\big[\log F(s_t)\big] \;+\; \sum_{k\ge 3}\frac{\kappa_k(s)}{k!}, \tag{81}$$

where $\kappa_k(s)$ is the k-th empirical cumulant of $\{X_v\}_{v\in V_s}$ (equivalently, of $\{\log F(s_t^{(v)})\}_v$, since $\log Z_\theta(q)$ is a visit-independent shift). The leading term is one-half the cross-visit variance of the log state-flow at occurrences of s.

Proof. The cumulant generating function admits the standard Taylor expansion

$$\log \mathbb{E}\big[e^{\lambda X}\big] \;=\; \sum_{k\ge 1}\frac{\kappa_k}{k!}\,\lambda^k \;=\; \kappa_1\lambda + \frac{\kappa_2}{2}\lambda^2 + \sum_{k\ge 3}\frac{\kappa_k}{k!}\lambda^k, \tag{82}$$

where $\kappa_1 = \mu = \mathbb{E}[X]$, $\kappa_2 = \sigma^2 = \mathrm{Var}[X]$, and $\{\kappa_k\}_{k\ge 3}$ are the higher cumulants. Apply this expansion to Eq. (75) (which expresses $\Lambda_\lambda^{(s)}$ as a CGF):

$$\Lambda_\lambda^{(s)} \;=\; G(s)\,\lambda + \frac{\mathrm{Var}_{V_s}[X]}{2}\,\lambda^2 + \sum_{k\ge 3}\frac{\kappa_k(s)}{k!}\,\lambda^k. \tag{83}$$

Setting $\lambda = 1$ and rearranging gives Eq. (81). The variance of X_v equals the variance of $\log F(s_t)$ over V_s since the additive constant $-\log Z_\theta(q)$ does not affect variance. □

Remark 6 (Stability Diagnostic via the Jensen Gap)**.** *A small Jensen gap $\Lambda_1^{(s)} - G(s)$ means X_v is approximately constant across the visits of s, i.e., $\log F(s_t)$ takes nearly the same value whenever s is invoked: s contributes* consistently *to flow regardless of context. A large gap conversely indicates that s's flow contribution varies substantially across contexts, signaling a* context-inconsistent *skill that may merit refining—the use case for the $\mathcal{U}$ class in main-text Eq. 13 (formal definition in Appendix F).*

Remark 7 (Centered Log-Flow Share)**.** *The centered log-flow share*

$$\widetilde{\Lambda}(s) \;\coloneqq\; \Lambda_1^{(s)} - \mathbb{E}_{s'}\big[\Lambda_1^{(s')}\big] \tag{84}$$

of main-text Eq. 13 subtracts the library-mean log skill marginal flow from $\Lambda_1^{(s)}$, ranking each skill's contribution relative *to the library. By Lemma 23, $\widetilde{\Lambda}(s)$ depends on s only through $\hat{F}(s)$ (the $\log Z_\theta(q)$ shift cancels), so $\widetilde{\Lambda}(s) = \log\hat{F}(s) - \mathbb{E}_{s'}[\log\hat{F}(s')]$ is a relative log-flow rank.*

E DAG Acyclicity Proof

Theorem 26 (Orchestration Graph is a DAG)**.** *Under the SkillFlow environment definition with three-way policy factorization (Equation* (5)*, main text) and frozen skill library within each training phase, the orchestration state graph $\mathcal{G}$ is a directed acyclic graph.*

Proof. Define a strict order on states by their history length: $\mathrm{depth}(s) := |H_s|$, i.e., the number of tokens in the interaction history at state s.

By the state-update rule (main-text Eq. (4)), $H_t = H_{t-1} \oplus (r_t, a_t, o_t^{\text{exec}})$ strictly appends three non-empty components to the history. Therefore:

$$\mathrm{depth}(s_t) = |H_t| > |H_{t-1}| = \mathrm{depth}(s_{t-1}). \tag{85}$$

For any directed edge $s \to s'$ in the orchestration graph, we have:

$$\mathrm{depth}(s') > \mathrm{depth}(s). \tag{86}$$

A cycle would be a path $s_0 \to s_1 \to \cdots \to s_k \to s_0$ with $k \geq 1$. Following the edges:

$$\text{depth}(s_1) > \text{depth}(s_0), \quad \text{depth}(s_2) > \text{depth}(s_1), \quad \ldots, \quad \text{depth}(s_0) > \text{depth}(s_k). \tag{87}$$

Combining: $\text{depth}(s_0) > \text{depth}(s_k) > \cdots > \text{depth}(s_0)$, a contradiction.

Therefore, no cycle exists, and $\mathcal{G}$ is a DAG. □

F Skill Curation Details

This appendix specifies (i) the phase-boundary detection rule, (ii) the curation classes $\mathcal{D}^-, \mathcal{R}, \mathcal{U}$ used by the operator Φ in main-text Eq. 13, (iii) the Skill Creator Ψ with its trigger steps, (iv) the resulting curation algorithm, and (v) the formal definition of *atomic composability* together with the proof that Φ preserves it.

F.1 Phase-Boundary Detection Rule

Within phase k, the squared TTB residual $\Delta(\tau)^2$ is tracked as a running mean over a sliding window of W training steps:

$$\overline{\Delta^2}_w^{(k)} \;\coloneqq\; \frac{1}{W} \sum_{i=w-W+1}^{w} \frac{1}{|\mathcal{B}_i|} \sum_{\tau \in \mathcal{B}_i} \Delta(\tau \mid \mathcal{S}^{(k)})^2, \tag{88}$$

where w is the current step within phase k and $\mathcal{B}_i$ is the mini-batch at step i. By Proposition 2 and the residual-floor identity $\bar{\Delta}^{*(k)} = \inf_\theta \mathbb{E}_\tau[\Delta(\tau \mid \mathcal{S}^{(k)}, \theta)^2]$ (main-text Eq. 12), $\overline{\Delta^2}_w^{(k)}$ asymptotes to a value $\geq \bar{\Delta}^{*(k)}$, and gradient descent halts further reduction once that floor is reached.

Definition 13 (Plateau Trigger)**.** *Let $\rho > 0$ be a relative-decrease tolerance and M a window-count budget. We say training has* plateaued *at step w within phase k if*

$$\frac{\overline{\Delta^2}_{w-W}^{(k)} \;-\; \overline{\Delta^2}_w^{(k)}}{\overline{\Delta^2}_{w-W}^{(k)}} \;<\; \rho. \tag{89}$$

Phase $k{+}1$ is triggered at the first step w for which Eq. (89) *holds for M consecutive non-overlapping windows.*

Concrete values for W, ρ, M are fixed throughout training and detailed in the supplementary code.

F.2 Curation Classes

At every triggered phase boundary $k \to k+1$, each existing skill $s \in \mathcal{S}^{(k)}$ is classified into one of three disjoint sets via the CGF statistics of Lemmas 22–24 (Appendix D.5). Let $\Phi^G_{\text{thr}}, \Phi^J_{\text{thr}} \in \mathbb{R}$ be hyperparameters; let $n^-(s)$ count the cumulative number of past phase boundaries at which $\widetilde{\Lambda}(s) < 0$.

Definition 14 (Curation Classes)**.** *The library is partitioned into three disjoint subsets:*

$$\mathcal{D}_k^- \;\coloneqq\; \big\{\, s \in \mathcal{S}^{(k)} \;:\; n^-(s) \geq K^- \big\}, \tag{90}$$

$$\mathcal{R}_k \;\coloneqq\; \big\{\, s \in \mathcal{S}^{(k)} \setminus \mathcal{D}_k^- \;:\; G(s) \geq \Phi^G_{thr},\; \Lambda_1^{(s)} - G(s) \leq \Phi^J_{thr} \big\}, \tag{91}$$

$$\mathcal{U}_k \;\coloneqq\; \mathcal{S}^{(k)} \setminus \big(\mathcal{D}_k^- \cup \mathcal{R}_k\big). \tag{92}$$

The three roles are: $\mathcal{D}_k^-$ prunes *skills with persistently negative centered share; $\mathcal{R}_k$* retains *skills with high mean log-flow and low Jensen gap; $\mathcal{U}_k$* refines *the remaining skills (high Jensen gap or low G).*

The Jensen gap threshold Φ^J_{thr} implements the stability diagnostic of Remark 6: a skill with consistent flow contribution across visits has small gap and is retained; a context-inconsistent skill has large gap and is refined. Concrete threshold values are detailed in the supplementary code.

F.3 Skill Creation from Success/Failure Trajectory Pairs

The Skill Creator Ψ is invoked at high-step-importance positions where successful and failed trajectories from the same query diverge.

Definition 15 (Trigger Steps)**.** *For each query q in the validation pool, let τ^+ be a successful trajectory ($R(\tau^+) = 1$) and τ^- a same-query failed trajectory ($R(\tau^-) = 0$) sampled under $\mathcal{S}^{(k)}$ with the current policy π_θ. The set of* trigger steps *for (q, τ^+, τ^-) is*

$$\mathcal{T}_q^{trig} \;\coloneqq\; \Big\{\, t \in \{1, \ldots, |\tau^+|\} \;:\; \log I(t)\big|_{\tau^+} \geq \zeta_{trig} \;\wedge\; t \notin \mathrm{cov}(\mathcal{R}_k \cup \mathcal{U}'_k) \Big\}, \tag{93}$$

where ζ_{trig} is a high-importance threshold and $\mathrm{cov}(\cdot)$ marks steps already covered by surviving (or refined) skills. By Lemma 14, $\log I(t)\big|_{\tau^+} \gg 0$ marks decisions whose quality became clear only under the hindsight backward—precisely the gap candidates for new tips.

Definition 16 (Skill Creator Ψ)**.** *The Skill Creator Ψ is a frozen LLM-based generator constrained to render* atomic tips *(Definition 18). Given creation context $c = (q, \tau^+, \tau^-, t)$ for each $t \in \mathcal{T}_q^{trig}$, Ψ outputs a textual atomic tip*

$$s_{new} \;=\; \Psi(c, \mathcal{T}, \mathcal{S}^{(k)}) \tag{94}$$

where $\mathcal{T}$ denotes the validation buffer of (q, τ^+, τ^-) trajectory pairs available at the phase boundary. The tip captures the strategic decision differentiating τ^+ from τ^- at step t. The output is constrained at the prompt level to be (a) self-contained (no reference to other tips at runtime), (b) of bounded textual length $L_{\max}$, and (c) phrased as strategic guidance (not as a literal action). Ψ also operates in a refine mode *on $\mathcal{U}_k$, rewriting context-inconsistent tips under the same atomic constraints.*

F.4 The Curation Operator Φ

Definition 17 (Evolution Operator)**.** *The CGF-based curation operator Φ produces the next-phase library by combining three sets:*

$$\mathcal{S}^{(k+1)} \;\coloneqq\; \Phi\big(\mathcal{S}^{(k)};\, \{(G(s), \widetilde{\Lambda}(s))\}_{s \in \mathcal{S}^{(k)}},\, \{\log I(t)\}_t\big) \;=\; \mathcal{R}_k \,\cup\, \mathcal{U}'_k \,\cup\, \Psi_k^{new}, \tag{95}$$

where $\mathcal{R}_k$ is the retained set (Definition 14); $\mathcal{U}'_k \coloneqq \{\Psi(s,\, \textit{refine}) : s \in \mathcal{U}_k\}$ is the refined set; and

$$\Psi_k^{new} \;\coloneqq\; \bigcup_{(q,\tau^+,\tau^-)} \;\bigcup_{t \in \mathcal{T}_q^{trig}} \big\{\Psi\big((q, \tau^+, \tau^-, t),\, \mathcal{T},\, \mathcal{S}^{(k)}\big)\big\} \tag{96}$$

is the newly-created tip set. Pruned tips $\mathcal{D}_k^-$ are absent from $\mathcal{S}^{(k+1)}$ by construction.

F.5 Atomic Composability and Its Preservation

Definition 18 (Atomic Tip)**.** *A skill s is an* atomic tip *if (i) s is a textual prompt fragment of bounded length $\leq L_{\max}$, (ii) the action $\texttt{skill}(s)$ deterministically appends s as a strategic-guidance segment to H_{t-1} without reading or modifying any other skill in $\mathcal{S}$ at runtime, and (iii) the cost of invoking s (in tokens and wall-clock) is bounded by a constant independent of $|\mathcal{S}|$.*

Definition 19 (Atomic Composability of a Library)**.** *A skill library $\mathcal{S}$ is* atomically composable *if every $s \in \mathcal{S}$ is an atomic tip (Definition 18).*

Lemma 27 (Φ Preserves Atomic Composability)**.** *If $\mathcal{S}^{(k)}$ is atomically composable, and Ψ is constrained to produce only atomic tips (Definition 16, in both creation and refine modes), then $\mathcal{S}^{(k+1)} = \Phi(\mathcal{S}^{(k)}; \ldots)$ is atomically composable.*

Proof. By Definition 17, $\mathcal{S}^{(k+1)} = \mathcal{R}_k \cup \mathcal{U}'_k \cup \Psi_k^{\text{new}}$. We verify atomicity for each constituent.

Retained skills ($\mathcal{R}_k \subseteq \mathcal{S}^{(k)}$). By the hypothesis on $\mathcal{S}^{(k)}$, every $s \in \mathcal{S}^{(k)}$ is atomic; the subset $\mathcal{R}_k$ inherits atomicity unchanged.

Refined skills ($\mathcal{U}'_k = \Psi(\mathcal{U}_k, \textbf{refine})$). By Definition 16, Ψ in refine mode is constrained to produce atomic tips. Each $s' \in \mathcal{U}'_k$ is therefore atomic.

New skills ($\Psi_k^{\textbf{new}}$). Each $s_{\text{new}} \in \Psi_k^{\text{new}}$ is the output of Ψ applied at a trigger step (Eq. (94)); by the same constraint it is atomic.

Pruned tips $\mathcal{D}_k^-$ are removed from $\mathcal{S}^{(k+1)}$ entirely. Atomicity is a per-tip structural property (Definition 18); it is preserved under set union, removal of subsets, and replacement of subsets by other atomic tips. Therefore every $s \in \mathcal{S}^{(k+1)}$ is atomic, and $\mathcal{S}^{(k+1)}$ is atomically composable. □

Remark 8 (Action-Space Constancy Within a Phase)**.** *Φ operates only at phase boundaries, so within phase k the library $\mathcal{S}^{(k)}$ is fixed. Combined with Lemma 27, this guarantees that the per-phase action space is constant and atomically composable, satisfying the structural prerequisite for TB-based training (Proposition 1, main text).*

F.6 Curation Algorithm

The full curation procedure at phase boundary $k \to k+1$ is summarized below.

Algorithm 1: Skill-Library Curation at Phase Boundary $k \to k+1$.

1. For each $s \in \mathcal{S}^{(k)}$, compute the per-skill CGF $\Lambda_\lambda^{(s)}$ at $\lambda \in \{0, 1\}$ over the recent batch $\mathcal{B}_s$ via the zero-cost formulas of Proposition 20.
2. Derive the summaries $G(s)$, $\Lambda_1^{(s)}$, and $\widetilde{\Lambda}(s) = \Lambda_1^{(s)} - \mathbb{E}_{s'}[\Lambda_1^{(s')}]$ via Lemmas 22, 23 and Remark 7.
3. Classify each $s \in \mathcal{S}^{(k)}$ into $\mathcal{D}_k^-$, $\mathcal{R}_k$, or $\mathcal{U}_k$ via Definition 14.
4. Refine each $s \in \mathcal{U}_k$ via Ψ in refine mode to produce $\mathcal{U}_k'$.
5. From the validation buffer, sample same-query success/failure pairs (τ^+, τ^-); identify trigger steps $\mathcal{T}_q^{\text{trig}}$ via Definition 15.
6. For each trigger step, invoke Ψ in creation mode to obtain new atomic tips Ψ_k^{new} (Eq. (96)).
7. Assemble $\mathcal{S}^{(k+1)} = \mathcal{R}_k \cup \mathcal{U}_k' \cup \Psi_k^{\text{new}}$ (Eq. (95)).
8. Warm-start π_θ and P_ϕ from phase k; reinitialize the partition function $Z_\theta(q)$ for the new action space.

By Lemma 27, this procedure preserves atomic composability across all phase transitions; together with Lemma 8, the post-evolution graph $\mathcal{G}$ remains a tree-structured DAG, satisfying the prerequisites for TB-based training within phase $k+1$.

Full prompt templates for Ψ (creation and refine modes) and complete hyperparameter values are provided in the supplementary code.

G Reward Function Details

$R(\tau)$ is an outcome-based scalar reward. For multi-hop QA tasks, $R(\tau) = \text{EM}(y_q, y^*)$ (exact match); for mathematical reasoning, $R(\tau) = \text{Acc}(y_q, y^*)$; and for code generation, $R(\tau) = \text{Pass@1}(y_q)$. All rewards lie in $[0, 1]$. We apply ε-smoothing (Appendix H) to ensure positive support.

H ε-Smoothing Analysis

When $R(\tau)$ can be zero (e.g., failed orchestration), Theorem 6 requires strictly positive rewards. We handle this via smoothing.

Definition 20 (ε-Smoothing)**.** *We define the smoothed reward as:*

$$\tilde{R}(\tau) \;\coloneqq\; R(\tau) + \varepsilon_{\min}, \tag{97}$$

where $\varepsilon_{\min} > 0$ is a small constant.

Proposition 28 (Ordering Preservation)**.** *Smoothing preserves the strict reward ordering: for any τ_1, τ_2 with $R(\tau_1) > R(\tau_2) \geq 0$:*

$$\tilde{R}(\tau_1)^\beta > \tilde{R}(\tau_2)^\beta, \tag{98}$$

hence the relative quality ranking of trajectories is preserved.

Proof. Since $R(\tau_1) > R(\tau_2) \geq 0$, we have:

$$\tilde{R}(\tau_1) = R(\tau_1) + \varepsilon_{\min} > R(\tau_2) + \varepsilon_{\min} = \tilde{R}(\tau_2). \tag{99}$$

Both $\tilde{R}(\tau_1), \tilde{R}(\tau_2) > 0$. Since $x \mapsto x^\beta$ is strictly monotone increasing on $\mathbb{R}_{>0}$ for $\beta > 0$:

$$\tilde{R}(\tau_1)^\beta > \tilde{R}(\tau_2)^\beta. \tag{100}$$

□

Proposition 29 (Flow Perturbation Bound)**.** *For $\beta \geq 1$, by the mean-value theorem applied to the convex function $x \mapsto x^\beta$ on $[R(\tau_x), \tilde{R}(\tau_x)]$,*

$$\left|\tilde{R}(\tau_x)^\beta - R(\tau_x)^\beta\right| \;\leq\; \beta\,\varepsilon_{\min}\,\max\!\big(\tilde{R}(\tau_x), R(\tau_x)\big)^{\beta-1}, \qquad \beta \geq 1. \tag{101}$$

For small $\varepsilon_{\min}$ the perturbation is negligible.

Remark 9 (Practical Choice of β)**.** *SkillFlow uses $\beta \geq 1$ in all reported experiments; the perturbation bound therefore always falls in the regime of Eq.* (101)*.*

I Gradient Variance Analysis

This section provides detailed analysis of the TTB gradient variance and compares it to standard policy-gradient methods.

I.1 TTB Self-Annealing Theorem with Chain Rule Expansion

Theorem 30 (TTB Gradient Self-Annealing)**.** *Treating $T = |\tau|$ as fixed in θ for each given τ, the gradient of the TTB loss with respect to θ satisfies:*

$$\nabla_\theta \mathcal{L}_{\mathrm{TTB}}(\tau) \;=\; \frac{2\,\Delta(\tau)}{T^2} \cdot \nabla_\theta \Delta(\tau). \tag{102}$$

Expanding $\nabla_\theta \Delta(\tau)$ by the chain rule on Definition 9:

$$\nabla_\theta \Delta(\tau) \;=\; \nabla_\theta \log Z_\theta(q) + \sum_{t=1}^{T} \nabla_\theta \widetilde{\log \pi}_\theta(a_t \mid r_t, H_{t-1}) \;-\; 0 \;-\; 0, \tag{103}$$

where the last two terms vanish because $\beta \log \tilde{R}(\tau)$ and $\widetilde{\log P}_\phi$ do not depend on θ.

Therefore:

$$\nabla_\theta \mathcal{L}_{\mathrm{TTB}}(\tau) \;=\; \frac{2\,\Delta(\tau)}{T^2} \cdot \left[\nabla_\theta \log Z_\theta(q) + \sum_{t=1}^{T} \nabla_\theta \widetilde{\log \pi}_\theta(a_t \mid r_t, H_{t-1})\right]. \tag{104}$$

As $\Delta(\tau) \to 0$, the prefine $2\Delta(\tau)/T^2$ shrinks and $\|\nabla_\theta \mathcal{L}_{\mathrm{TTB}}\| \to 0$ for every fixed-length trajectory; trajectories of different T contribute on a comparable scale because of the $1/T^2$ length normalizer.

Proof. Apply the chain rule to $\mathcal{L}_{\mathrm{TTB}}(\tau) = (\Delta(\tau)/T)^2 = \Delta(\tau)^2/T^2$:

$$\nabla_\theta \mathcal{L}_{\mathrm{TTB}}(\tau) \;=\; \nabla_\theta \left[\frac{\Delta(\tau)^2}{T^2}\right] \;=\; \frac{2\,\Delta(\tau)}{T^2}\,\nabla_\theta \Delta(\tau), \tag{105}$$

where T is determined by τ (not by θ) and is constant under ∇_θ. Since T varies across trajectories, $1/T^2$ remains a per-trajectory length normalizer. □

I.2 Variance Bound and Comparison to REINFORCE

Proposition 31 (Variance Bound under Bounded Gradient)**.** *Assume the residual gradient is uniformly bounded along training: there exists $G < \infty$ such that $\|\nabla_\theta \Delta(\tau)\| \le G$ for all τ, and assume $T \ge T_{\min}$ for a fixed $T_{\min} \ge 1$. Then the per-trajectory TTB gradient norm satisfies*

$$\big\|\nabla_\theta \mathcal{L}_{\mathrm{TTB}}(\tau)\big\| \;=\; \frac{2|\Delta(\tau)|}{T^2}\,\|\nabla_\theta \Delta(\tau)\| \;\le\; \frac{2G\,|\Delta(\tau)|}{T_{\min}^2}, \tag{106}$$

which yields the variance bound

$$\mathrm{Var}_\tau\big[\nabla_\theta \mathcal{L}_{\mathrm{TTB}}(\tau)\big] \;\le\; \mathbb{E}_\tau\big[\|\nabla_\theta \mathcal{L}_{\mathrm{TTB}}(\tau)\|^2\big] \;\le\; \frac{4G^2}{T_{\min}^4}\,\mathbb{E}_\tau\big[\Delta(\tau)^2\big]. \tag{107}$$

Proof. Eq. (106) follows from Theorem 30 together with the bounded-gradient assumption. Squaring and taking expectation gives Eq. (107); the variance bound uses $\mathrm{Var} \le \mathbb{E}[\|\cdot\|^2]$. □

As training converges, $\mathbb{E}_\tau[\Delta(\tau)^2] \to 0$ and Eq. (107) forces $\mathrm{Var}_\tau[\nabla_\theta \mathcal{L}_{\mathrm{TTB}}] \to 0$.

In contrast, the REINFORCE estimator $g_{\mathrm{PG}}(\tau) \coloneqq (R(\tau) - b)\,\nabla_\theta \log \pi_\theta(\tau)$ has

$$\mathrm{Var}_\tau[g_{\mathrm{PG}}(\tau)] \;=\; \mathrm{Var}_\tau\big[(R(\tau) - b)\,\nabla_\theta \log \pi_\theta(\tau)\big], \tag{108}$$

which is *not forced* to vanish at convergence: it can stay strictly positive whenever the converged policy remains stochastic and sampled trajectories carry heterogeneous rewards (Lemma 32 below). Vanishing only occurs in the special cases of a deterministic optimal policy or rewards being identical across all sampled trajectories.

Lemma 32 (REINFORCE Variance Is Not Forced to Vanish)**.** *In REINFORCE, gradient estimates are $(R(\tau)-b)\,\nabla_\theta \log \pi_\theta(\tau)$, where b is a baseline. Whenever the converged policy remains stochastic and the sampled trajectories carry heterogeneous rewards, $\mathrm{Var}_\tau[R(\tau)] > 0$ and the estimator's variance has a strictly positive lower bound: it is* not forced *to vanish at convergence. (Vanishing can occur in the special cases of a deterministic optimal policy, or rewards being identical across all sampled trajectories.)*

By contrast, TTB uses a regression loss whose residual Δ itself is the target for convergence. When $\Delta(\tau) \to 0$ pointwise, Theorem 30 forces the gradient norm $\|\nabla_\theta \mathcal{L}_{\mathrm{TTB}}\| \to 0$; under the bounded-gradient assumption of Proposition 31, the variance also vanishes.

J Proof of Proposition 2 (Main Text)

Proposition 2 (main text): TTB training induces reward-proportional sampling and yields per-step credit at no extra inference cost.

We prove a slightly more detailed version: at convergence the gradient variance vanishes (residual $\to$ 0), the learned policy samples trajectories in proportion to tempered reward, and the step importance ratio gives a multiplicative per-step decomposition of trajectory-level credit.

Proof. (i) TTB gradient variance vanishes.

By Theorem 30, $\nabla_\theta \mathcal{L}_{\mathrm{TTB}}(\tau) = (2\Delta(\tau)/T^2)\cdot\nabla_\theta \Delta(\tau)$. As $\Delta(\tau) \to 0$ during training (which occurs at the optimum by the definition of the loss), the factor $\Delta(\tau)/T$ shrinks, causing $\|\nabla_\theta \mathcal{L}_{\mathrm{TTB}}\| \to 0$. By Proposition 31, both variance terms $\mathbb{E}[\Delta^2]$ and $\mathrm{Var}[\Delta]$ vanish, so $\mathrm{Var}[\nabla_\theta \mathcal{L}_{\mathrm{TTB}}] \to 0$.

In contrast, by Lemma 32, REINFORCE and GRPO gradients have variance proportional to reward variance, which persists at convergence.

(ii) Policy samples in proportion to tempered reward.

At convergence $\Delta(\tau) = 0$ for all τ, so the TB condition (Theorem 9) is satisfied with reward $\tilde{R}(\tau)^\beta$. By the GFlowNet sampling theorem (Theorem 6), the conditional action-sequence distribution then satisfies $\pi_\theta(a_{1:T} \mid r_{1:T}, o^{\mathrm{exec}}_{1:T}, q) \propto \tilde{R}(\tau)^\beta$ (matching main-text §4.2); marginalising over reasoning and execution context recovers the unconditional form $\pi_\theta(\tau \mid q) = \tilde{R}(\tau)^\beta / Z_\theta(q)$.

(iii) Step importance provides per-step credit decomposition.

By Corollary 19, the step importance is:

$$I(t) = \frac{F(s_t)}{F(s_{t-1})} = \frac{\pi_\theta(a_t \mid r_t, H_{t-1})}{P_\phi(a_t \mid H_{t-1} \oplus o_t^{\text{exec}})}. \tag{109}$$

This decomposes the trajectory-level flow $F(\tau)$ multiplicatively:

$$F(\tau) = Z_\theta(q) \cdot \prod_{t=1}^{T} I(t). \tag{110}$$

Each factor $I(t)$ quantifies the amplification of flow at decision t, providing an interpretable per-step attribution. Unlike Monte-Carlo rollout baselines (which require additional samples), $I(t)$ is computed from the forward and backward policies already evaluated in the TTB loss. □

K Proof of Proposition 3 (Main Text)

Proposition 3 (main text): Flow-driven recursive evolution autonomously expands the skill library while preserving its atomic composability.

We prove a slightly more detailed version with three parts: (i) saturation of the running TTB residual against the library-conditional floor $\bar{\Delta}^{*(k)}$ is a sufficient diagnostic of joint expressiveness limits and triggers phase $k \to k{+}1$; (ii) the curation operator Φ preserves atomic composability; (iii) training within a phase keeps flow conservation under the frozen library.

Proof. (i) Plateau saturation is a sufficient diagnostic of joint expressiveness limits.

Within training phase k, the per-trajectory squared residual $\Delta(\tau \mid \mathcal{S}^{(k)}, \theta, \phi, Z_\theta)^2$ enters the TTB loss as $\mathcal{L}_{\text{TTB}}(\tau) = (\Delta/T)^2$ (Definition 10). By Theorem 30 the gradient drives $\Delta(\tau)^2 \to 0$ on every trajectory subject to the expressiveness of the parameterization. The infimal expected squared residual under the current library is the floor $\bar{\Delta}^{*(k)}$ (main-text Eq. 12):

$$\bar{\Delta}^{*(k)} \;:=\; \inf_{\theta,\phi,Z_\theta} \mathbb{E}_\tau\big[\Delta(\tau \mid \mathcal{S}^{(k)}, \theta, \phi, Z_\theta)^2\big] \;\geq\; 0. \tag{111}$$

The two directions are asymmetric:

(Sufficiency, $\Rightarrow$) If the running mean $\overline{\Delta^2}_w^{(k)}$ (Eq. (88)) saturates—i.e., its relative decrease across M consecutive windows falls below the tolerance ρ (Eq. (89))—then under standard SGD descent assumptions $\overline{\Delta^2}_w^{(k)}$ has approached $\bar{\Delta}^{*(k)}$. If $\bar{\Delta}^{*(k)} > 0$, this directly evidences that the joint ($\mathcal{S}^{(k)}$, policy class, P_ϕ class, Z_θ family) cannot represent the reward-matching TB condition; phase $k+1$ is triggered, and skill evolution attributes this insufficiency *a fortiori* to $\mathcal{S}^{(k)}$, expanding the action space to enable further residual reduction.

(Necessity, $\Leftarrow$, only as a heuristic) Plateau saturation is *not* an exclusive signature of library inadequacy: a saturated $\overline{\Delta^2}_w^{(k)}$ may also reflect limited policy or backward-policy capacity, an under-trained partition function, exploration deficits, optimizer stagnation, or finite-batch noise. We therefore treat the plateau as a *sufficient diagnostic* for triggering evolution, with the implicit operating assumption that other capacity bottlenecks have been controlled by standard practice (warm-start, learning-rate schedules, replay buffers).

This is the precise sense in which the stagnation criterion "triggers" skill evolution: it provides a sufficient signal under controlled conditions, not an iff-equivalence with library inadequacy alone.

(ii) The CGF-based curation operator Φ preserves atomic composability.

By Lemma 27 (Appendix F.5), if $\mathcal{S}^{(k)}$ is atomically composable (Definition 19) and the Skill Creator Ψ is constrained to produce atomic tips in both creation and refine modes (Definition 16), then $\mathcal{S}^{(k+1)} = \Phi(\mathcal{S}^{(k)}; \{(G(s), \widetilde{\Lambda}(s))\}_s, \{\log I(t)\}_t)$ is atomically composable. The base case $\mathcal{S}^{(0)}$ is

atomically composable by initialization (the seed library consists of bounded-length, self-contained tips). Induction on k then yields atomic composability of $\mathcal{S}^{(k)}$ for every phase $k \geq 0$. The CGF inputs $G(s)$ and $\widetilde{\Lambda}(s)$ used to drive Φ's classification are well-defined (Lemmas 22–23) and Φ acts only by partitioning $\mathcal{S}^{(k)}$ and adjoining Ψ's atomic outputs; no operation introduces non-atomic tips.

(iii) Frozen libraries within a phase guarantee flow conservation.

Within phase k, the library $\mathcal{S}^{(k)}$ is fixed (Remark 8); the environment $\mathcal{E}^{(k)}$, action space, and DAG structure $\mathcal{G}^{(k)}$ are therefore fixed. By Theorem 26 (DAG acyclicity) and Lemma 18 (tree-DAG uniqueness), $\mathcal{G}^{(k)}$ admits a unique reward-matching flow F when terminal values $\{F(x) = \tilde{R}(\tau_x)^\beta\}_{x \in \mathcal{X}}$ are prescribed. Training π_θ and P_ϕ to satisfy the SkillFlow TB condition (Eq. (49), Lemma 14) is equivalent to solving for that flow (Theorem 9). At convergence, flow conservation holds at every non-terminal state:

$$F(s) \;=\; \sum_{s' \in \mathrm{Ch}(s)} F(s \to s') \;=\; \sum_{s' \in \mathrm{Pa}(s)} F(s' \to s), \tag{112}$$

which is the defining property of a valid flow (Definition 2).

At the phase transition $k \to k+1$, Φ produces $\mathcal{S}^{(k+1)}$ (Definition 17); the new graph $\mathcal{G}^{(k+1)}$ retains its tree-DAG structure by Lemma 8 (strict history growth is unaffected by library change). To smooth the transition we warm-start π_θ, P_ϕ from phase k and reinitialize $Z_\theta(q)$ for the expanded action space (Algorithm 1, Step 8). Within phase $k+1$, the same reasoning yields flow conservation at convergence. By induction on k, flow conservation is guaranteed within every phase. □

L Backward Policy Design and Implementation

This section details the design principles and implementation of the backward policy P_ϕ.

L.1 Hindsight Conditioning

The key insight is that P_ϕ conditions on information unavailable to the forward policy:

Definition 21 (Hindsight-Enriched State)**.** *The hindsight-enriched state at step t is:*

$$H_{t-1}^{\textit{hindsight}} \;\coloneqq\; H_{t-1} \oplus o_t^{\textit{exec}}, \tag{113}$$

which augments the forward state H_{t-1} with the execution observation $o_t^{\textit{exec}}$ that π_θ could not access when selecting a_t.

This information asymmetry is what makes step importance meaningful:

Lemma 33 (Information Asymmetry and Credit Signal)**.** *At step t, the forward policy π_θ selects action a_t given H_{t-1} (before execution). The backward policy P_ϕ evaluates the same action given $H_{t-1} \oplus o_t^{\textit{exec}}$ (after execution).*

If $P_\phi(a_t \mid H_{t-1}^{\textit{hindsight}}) \ll \pi_\theta(a_t \mid H_{t-1})$, the action appeared good before execution but turned out poorly in hindsight. If $P_\phi(a_t \mid H_{t-1}^{\textit{hindsight}}) \gg \pi_\theta(a_t \mid H_{t-1})$, the action was initially uncertain but validated by execution.

The ratio $I(t) = \pi_\theta / P_\phi$ captures this update in assessment, providing a meaningful credit signal without additional samples.

L.2 Think-Action Separation

To focus flow balance on the decision space rather than reasoning verbosity:

Definition 22 (Action Token vs. Reasoning Token)**.** *Each step t contains:*

- ***Reasoning tokens**: Free-form text in r_t for chain-of-thought.*
- ***Action tokens**: Structured JSON in $a_t = (\alpha_t, o_t)$ specifying the decision.*

Only action tokens contribute to log-probabilities in π_θ and P_ϕ:

$$\log \pi_\theta(a_t \mid H_{t-1}) = \sum_{j \in \mathcal{A}_t} \log \pi_\theta(\textit{token}_j \mid \textit{token}_{<j}, H_{t-1}), \tag{114}$$

where $\mathcal{A}_t$ denotes the set of positions of action tokens at step t. Reasoning tokens are part of the context; their content influences future decisions but does not participate in the flow balance.

This separation ensures that flow balance operates on the decision quality, not reasoning length.

L.3 Implementation Details

Architecture P_ϕ shares the base LLM (Qwen3.5-9B) with π_θ but uses a separate LoRA adapter. Sharing the base model reduces memory and computation; LoRA adapters are swapped at inference time by named adapter selection.

Adapter Configuration ϕ-LoRA uses rank 32, targeting $\{q, v\}$ projections in attention layers. This is smaller than the θ-LoRA (rank 64, targeting $\{q, k, v, o\}$).

Token-Level Conditioning P_ϕ receives the hindsight-enriched state as context, then evaluates the log-probability of action tokens autoregressively. This respects the causal structure: earlier action tokens do not condition on later ones.

M Dataset Details

We evaluate SkillFlow on 14 public benchmarks covering four task categories: question answering, mathematical reasoning, interactive decision making, and code generation. Seven datasets are used as in-distribution (IID) benchmarks for training and evaluation; the remaining seven are held out as out-of-distribution (OOD) generalization tests, with the skill library frozen at end-of-training.

M.1 In-Distribution Datasets

- **HotpotQA** [Yang et al., 2018]: A large-scale multi-hop QA corpus where each question requires reasoning across multiple Wikipedia paragraphs to derive the answer.
- **TriviaQA** [Joshi et al., 2017]: A reading-comprehension dataset of trivia question–answer pairs paired with evidence documents.
- **MedQA** [Jin et al., 2021]: A multiple-choice medical QA benchmark drawn from professional medical-licensing examinations, requiring multi-step clinical reasoning.
- **AIME 2026**: Problems from the 2026 American Invitational Mathematics Examination, used as a difficult mathematical-reasoning benchmark.
- **WebShop** [Yao et al., 2022a]: A simulated e-commerce environment where the agent interprets a natural-language instruction, navigates web pages, and selects the matching product.
- **ALFWorld** [Shridhar et al., 2020]: A text-based interactive environment grounded in household tasks; the agent issues actions in language and receives observation feedback.
- **SWE-bench** [Jimenez et al., 2023]: A benchmark of real-world GitHub issues that require generating code patches whose application resolves the issue.

M.2 Out-of-Distribution Datasets

- **MuSiQue** [Trivedi et al., 2022]: A composite multi-hop QA dataset built by chaining single-hop questions, designed to stress reasoning composition.
- **NQ-Open**: An open-domain QA derivative of Natural Questions, where the model must produce free-form answers from user-issued queries.
- **MATH-Hard**: The hard subset of competition mathematics problems (algebra, geometry, number theory, etc.), evaluated by exact-match correctness on the final answer.
- **GPQA Diamond**: The most challenging split of GPQA, containing graduate-level science questions across physics, chemistry, and biology.

- **HumanEval** [Chen et al., 2021]: A code-generation benchmark of hand-written Python problems with hidden unit tests; correctness is evaluated by execution.
- **ScienceWorld**: A text-based interactive science environment that requires the agent to perform multi-step experiments and reason over physical-world dynamics.
- **Mind2Web**: A web-navigation benchmark where the agent performs multi-step actions across real-world websites to complete user tasks.

N Baseline Details

We compare SkillFlow against four categories of baselines. Unless otherwise stated, all baselines that involve fine-tuning or reinforcement learning use the same Qwen3.5-9B backbone and the same training data as SkillFlow, isolating the orchestration objective and skill-evolution mechanism as the source of any performance gap.

N.1 Direct-LLM Baselines

- **Qwen3.5-9B**: The same backbone we use as Supervisor, queried directly without any orchestration training, providing a faithful no-orchestration reference point.
- **v4-flash**: A strong proprietary instruction-tuned LLM, queried in a single-turn ReAct-style prompting setup as a high-capacity baseline.
- **Claude Haiku 4.5**: Another high-capacity proprietary baseline used to bound how much of SkillFlow's gain arises from orchestration versus raw model strength.

N.2 Fine-Tuning Baselines

- **SFT (Qwen3.5-9B)**: Supervised fine-tuning on demonstration trajectories of orchestration plans, with no exploration or reward feedback.
- **GRPO (Qwen3.5-9B)** [Shao et al., 2024]: Group Relative Policy Optimization on the same backbone, using terminal rewards and the same initial skill library as SkillFlow.

N.3 Search-Based Workflow Baselines

- **AFlow** [Zhang et al., 2024]: A workflow-search method that explores compositions of pre-defined operators using MCTS guided by an LLM judge, with no parameter learning.

N.4 RL Agent Baselines

- **AgentFlow** [Li et al., 2025b]: An in-the-flow agentic system that jointly optimizes planning and tool use under reinforcement learning over multi-turn interactions.
- **FlowSteer** [Zhang et al., 2026b]: An end-to-end RL framework for interactive agentic workflow orchestration on a fixed skill library.
- **SkillRL** [Xia et al., 2026]: A skill-augmented RL agent that grows its action space using heuristic skill-distillation triggers, representative of the static-library RL paradigm.

O Evaluation Metrics

We adopt task-appropriate evaluation metrics consistent with each benchmark's standard protocol.

F1 Score. For QA-style benchmarks (HotpotQA, TriviaQA, MuSiQue, NQ-Open), we report the token-level F1 between the predicted answer y_i and the ground truth y_i^* after standard text normalization:

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{115}$$

where $\text{Precision} = |\text{tok}(y_i) \cap \text{tok}(y_i^*)|/|\text{tok}(y_i)|$ and $\text{Recall} = |\text{tok}(y_i) \cap \text{tok}(y_i^*)|/|\text{tok}(y_i^*)|$.

Exact Match (EM) / Accuracy. For mathematical reasoning (AIME, MATH-Hard) and multi-choice QA (MedQA, GPQA Diamond), we report exact-match accuracy after canonicalization:

$$\text{Acc} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\big(\text{norm}(y_i) = \text{norm}(y_i^*)\big), \tag{116}$$

where $\text{norm}(\cdot)$ standardizes whitespace, casing, and final-answer extraction.

Average Score and Success Rate. For interactive decision-making (WebShop, ALFWorld, ScienceWorld), we follow each environment's standard evaluator: *Average Score* reports the environment-defined task-specific score, while *Success Rate (SR)* is the fraction of episodes terminating in a fully solved state.

Resolved Rate. For SWE-bench [Jimenez et al., 2023], we report the fraction of issues whose generated code patch passes the held-out test suite (the standard "resolved" metric).

pass@1. For HumanEval [Chen et al., 2021], we report *pass@1*, the fraction of problems for which the first generated program passes all hidden unit tests, computed by sampling one program per problem with deterministic decoding.

Step-Level Metrics for Web Navigation. For Mind2Web, we report *Step Accuracy* (fraction of steps with correctly predicted target element and action) and *Action F1* (token-level F1 over the predicted action string against the ground truth).

P Computational Resources

We report the hardware and the main-run wall-clock and GPU-hour cost of training SkillFlow on Qwen3.5-9B.

P.1 Hardware

All training and on-policy rollout were performed on a single server with $4\times$ NVIDIA A100-SXM4 (80 GB) GPUs, 32 logical CPU cores, 549 GB RAM, and 1.6 TB local NVMe. Software stack: CUDA 12.1, PyTorch 2.4 with DeepSpeed and PEFT for training, vLLM 0.5 for the executor, and SGLang for the supervisor.

P.2 Main run: SkillFlow on Qwen3.5-9B

Wall-clock 73 h, ≈ 292 GPU-hours, 250 training steps (LoRA checkpoint saved every 10 steps, 25 checkpoints in total).

Item	Value
Base model	Qwen3.5-9B-Instruct
LoRA θ (forward)	rank 64, $\alpha = 128$, on q, k, v, o_proj
LoRA ϕ (backward)	rank 16, $\alpha = 32$, on q, v_proj
Z head (partition fn)	scalar parameter, separate optimizer
Optimizer	AdamW (default β_1, β_2); three optimizers (θ, Z, ϕ)
Learning rate	1.0×10^{-4}
Max grad norm	3.0
KL coeff ($\text{KL}(\pi_\theta \Vert \pi_{\text{ref}})$)	0.01
TTB temperature β	1.0
$\varepsilon_{\min}$	0.1
Effective batch	$28 = 7$ questions $\times$ 4 trajectories
Max episode length T	12
Mean step time	≈ 14.8 min/step (rollout + update + LoRA hot-swap)
Peak VRAM (per GPU)	≈ 70 GB / 80 GB (gradient checkpointing on)

Q Case Studies

This appendix presents five complementary case studies illustrating SkillFlow's behaviour at different granularities: training dynamics across phases (Q.1), library-level boom-and-prune cycles (Q.2), three real evolved skills with signal attribution to claims C1–C3 (Q.3), per-step importance signals on a successful trajectory (Q.4), and multi-trajectory success/failure comparison (Q.5). Numerical values are representative of one Qwen3.5-9B run; raw logs and trajectory dumps are released with the supplementary code.

Q.1 Training Dynamics: A Four-Phase Trajectory

Table 3 samples eight representative steps spanning the full 250-step training run. Four phases are visible:

1. **Bootstrap** (steps 0–25): the skill library is empty (WS$=0$); only the base policy drives reward, and $\mathcal{L}_{\text{TTB}}$ falls steeply as Z_θ adjusts.
2. **Emergence** (steps 25–75): the first plateau on $\mathcal{L}_{\text{TTB}}$ triggers the curation operator Φ, which begins generating skills (WS grows $0 \to 14$). Reward variance is high but $\log Z_\theta$ keeps rising.
3. **Maturity** (steps 75–175): the boom-and-prune cycle (P.2) operates; WS oscillates between 8 and 14 as $\hat{F}(s)$ drives prune/refine decisions.
4. **Steady state** (steps 175–250): WS stabilises around 11; flow entropy stays above 3.0, indicating that reward-proportional sampling preserves multiple high-reward sub-trajectories rather than collapsing to a single mode.

Step	$\mathcal{L}_{\text{TTB}}$	avg. R	avg. $\hat{y}$	avg. $\lvert\tau\rvert$	flow ent.	$\log Z_\theta$	WS	Phase
0	0.83	0.55	0.50	7.6	3.17	-2.30	0	Bootstrap
15	0.42	0.65	0.58	7.5	3.05	-2.05	0	Bootstrap
30	0.18	0.72	0.65	7.8	3.06	-1.65	6	Emergence
60	0.15	0.76	0.69	7.9	3.10	-1.55	12	Emergence
100	0.10	0.81	0.74	7.4	3.13	-1.50	9	Maturity
150	0.07	0.83	0.76	7.6	3.09	-1.45	11	Maturity
200	0.06	0.84	0.78	7.7	3.12	-1.42	10	Steady
250	0.05	0.85	0.80	7.6	3.17	-1.40	11	Steady

Table 3: Training dynamics on Qwen3.5-9B (eight representative steps from a 250-step run). $\mathcal{L}_{\text{TTB}}$: TTB loss; avg. R: average reward; avg. $\hat{y}$: average answer-correctness rate; avg. $|\tau|$: average trajectory length; flow ent.: $-\sum_a \pi_\theta(a) \log \pi_\theta(a)$ at terminal step; WS: skill-library size.

Q.2 Skill Library Evolution: Boom-and-Prune Cycles

The library size traces a characteristic boom-and-prune cycle rather than monotonically growing. Table 4 shows snapshots at eight phase boundaries; two "boom" peaks (WS$=22$ at step 50; WS$=18$ at step 130) are followed by single-step prune sweeps that remove 14 and 8 skills respectively. The mature library settles around 11 active skills covering all seven IID task categories.

Boom mechanism. A boom is the result of a single Φ invocation at a phase boundary: when the running mean of $\Delta(\tau)^2$ saturates against $\bar{\Delta}^{*(k)}$ (Eq. 12), the curator drains the accumulated (τ^+, τ^-) pair buffer at high-$|\log I(t)|$ steps and asks Ψ to synthesise candidate tips for every uncovered decision gap. Early phases accumulate the largest buffers because the policy is still exploring, which is why peak #1 (step 50, WS$=22$) is larger than peak #2 (step 130, WS$=18$).

Prune mechanism and convergence. Newly created skills enter the library with reset $\hat{F}(s)$; on the next batch the centred log-flow share $\widetilde{\Lambda}(s) = \Lambda_1^{(s)} - \mathbb{E}_{s'}[\Lambda_1^{(s')}]$ flags those that fail to attract flow and removes them (-14 after peak #1, -8 after peak #2). Once the policy class is expressive enough for the current task mix, $\widetilde{\Lambda}(s)$ stops promoting new candidates and the library stabilises: steps 195 and 250 show *identical* 11-skill compositions, indicating the minimum sufficient set under the trained π_θ. Cumulative creation across the run (≈ 63 skills) is $\approx 5.7\times$ the final library size, evidencing that

aggressive over-creation followed by flow-driven pruning is more effective than monotonic growth and confirming the $\hat{F}(s)$-driven design of §4.3.

Step	WS	Per-task distribution	Event
25	5	1 each: code, math, alfworld, webshop, factual	First Φ trigger
50	22	alfworld=6, math=5, webshop=4, multi=3, code=2, factual=2	Boom (peak #1)
55	8	alfworld=2, math=2, webshop=2, code=1, multi=1	Prune ($\Delta = -14$)
90	12	alfworld=3, math=2, webshop=2, multi=2, code=1, factual=1, science=1	Mid-training
130	18	alfworld=5, math=4, webshop=3, multi=2, code=2, factual=1, science=1	Boom (peak #2)
137	10	alfworld=2, math=2, webshop=2, multi=1, code=1, factual=1, science=1	Prune ($\Delta = -8$)
195	11	alfworld=2, math=2, webshop=2, multi=2, code=1, factual=1, science=1	Stable
250	11	alfworld=2, math=2, webshop=2, multi=2, code=1, factual=1, science=1	Final

Table 4: Skill-library snapshots at phase boundaries (all per-task sums match WS exactly). Cumulative creation count over the run: alfworld ≈ 16, math ≈ 12, webshop ≈ 11, multi-hop ≈ 8, factual ≈ 7, code ≈ 6, science ≈ 3 – harder, longer-horizon tasks attract more creation activity, consistent with the $\hat{F}(s)$-driven curation in §4.3. Snapshot timestamps are chosen at phase boundaries; intermediate steps sampled in Table 3 (e.g., t=150 with WS$=11$) lie between boom-prune events and reflect post-curation states.

Q.3 Real Evolved Skills with Signal Attribution

Beyond aggregate statistics, the most direct evidence of SkillFlow's contribution is the *content* of the skills it produces. We show three library members with the highest mean log-flow $G(s)$ at the end of training; each is paired with the flow-signal that drove its emergence and the paper claim it evidences.

Skill A: `tip-webshop` — success 78.5% over 200 invocations — claim C1 (TTB diversity preservation)

Curation trigger. $I(t)$ flagged `click[back_to_search]` as a high-importance step that almost always coincided with terminal R=0; Φ created the skill from the resulting success/failure pair set.

Rule Zero — never go back. Once you leave search results, you are committed. NEVER `click[back_to_search]`: every trajectory using it scored R=0.0.

Option selection. If instruction says "pink" and options are [pink | pink light blue | pink purple], click *only* "pink". Each click in the same category *overwrites* the previous selection (clicking "pink" then "pink light blue" → bought "pink light blue", scoring 0.667 instead of 1.0).

Counter-intuitive trade-off. If a product has no matching options, `click[buy_now]` anyway: a partial match ($R \in [0.03, 0.75]$) beats a back-to-search loop (R=0).

Why baselines miss this. The trade-off "buy partial $\succ$ loop" is unreachable from REINFORCE: it requires keeping a 0.03-reward trajectory alive in the batch long enough to discover it dominates a 0.0 loop. Reward-proportional sampling preserves this low-but-non-zero mode; mode-collapsing baselines never see it.

Skill B: `tip-alfworld-desklamp` — success 42.5% over 212 invocations — claim C2 (zero-cost per-step credit)

Hidden game-engine rule discovered. The ALFWorld documentation does not state this; SkillFlow extracted it from per-step credit signals on success/failure trajectory pairs.

Win condition. You are *holding the target item and the desklamp is on*. The task auto-completes *instantly* when both are true. No manual completion command exists, and `examine X` *never* triggers completion.

Fatal mistakes (from evidence). (i) `move X to Y` drops the item and breaks the win condition. (ii) `examine X with desklamp 1` after lamp-on never completes (5−7 wasted retries observed in failed trajectories). (iii) `go to desklamp 1` fails: desklamp is not a navigable location (4+ wasted retries).

⇒ **You do NOT need the item and the lamp at the same location.**

Why baselines miss this. The hidden rule surfaces because P_ϕ, conditioned on the post-execution observation, assigns identical hindsight log-probabilities to `take` actions *regardless of the agent's location*, while wasted `examine` / repeated `go to` actions receive vanishing $\log I(t)$. The backward policy thus localises which decisions were actually responsible for the reward — something terminal-only baselines cannot recover.

Skill C: `tip-factual-qa` — success 77.2% over 224 invocations — claim C3 (flow-driven curation)

Quantified by sibling-query A/B comparisons sampled in the same batch.

Always search before answering. Skipping search yields $R \approx 0.05$; searching first yields $R \geq 1.13$.

Query construction — reward gap up to 1.07 between siblings. Lead with the most distinctive entity. Example: `''TV detective George Toolan DS''` beats `''DS George Toolan TV detective''`. Include concrete numbers/units; do *not* pre-commit to a wrong candidate name in the query — this poisons retrieval.

Hard rules. Never use `lookup` (returns `NO_MATCH`; R=0.16 at 6 steps vs R=1.13 at 3 steps). On a `[REPEATED]` search, stop and `answer` immediately.

Why baselines miss this. Sibling-query reward gaps are observable only when both token-orderings appear in the same batch — a direct consequence of reward-proportional sampling. $I(t)$ then localises the search step as the high-importance decision, and $\hat{F}(s)$ ranks "query-construction" as a high-flow skill family worth retaining and refining.

Take-away. The three skills cover three distinct emergence modes: explicit forbidden actions from (τ^+, τ^-) pairs (A), hidden environment dynamics from per-step credit (B), and micro-decision sensitivity from in-batch siblings (C) — together directly instantiating the three SkillFlow claims.

Q.4 Per-Step Importance Signals: An ALFWorld Trajectory

Table 5 traces a successful 9-step trajectory τ^+ on the ALFWorld task "*put two watches on shelf*" (reward $\tilde{R}(\tau^+) = 0.86$). The step importance $I(t)$ separates two regimes:

- **Confirmed steps** ($I(t) \ll 1$, marked $\diamond$): forward and backward policies agree, indicating routine deterministic moves whose role becomes obvious after the action is taken.
- **Critical steps** ($I(t) \gg 1$, marked $\star$): the forward policy chose a low-prior action that the hindsight backward strongly endorses — these are the decisions that drove success.

The four high-$I(t)$ steps $(3, 4, 6, 8)$ cleanly identify the *navigate–pickup–place* pattern that distinguishes τ^+ from same-query failure trajectories.

Critical-step interpretation. The two highly-critical ($\star\star$) decisions `take watch 1` (step 4) and `move watch 1 to shelf 1` (step 6) are both *state-changing physical interactions* rather than navigation. Their backward log-probabilities ($\log P_\phi \approx -14$) are an order of magnitude below their forward values, reflecting that, conditional on the post-execution observation, hindsight assigns very high credit to actions that actually advanced the world state toward the goal. The single-star navigation steps connect these criticals into the `navigate`→`pickup`→`place` chain, while step 2 (`go to shelf 1`, $I(t) = 0.001$, marked $\diamond$) flags a wasted early navigation — exactly the kind of redundancy that the `tip-alfworld-routing` skill (Q.3, Skill B) is designed to prevent.

Telescoping closure. The cumulative log-flow $\log F(H_t)$ telescopes from $\log Z_\theta(q) = -2.30$ at t=0 to $+47.95$ at the terminal step, matching the sum $\sum_{t' \leq t} \log I(t')$ (Eq. 11) to within rounding. This trajectory-level identity is the empirical analogue of the Detailed-Balance condition $F(H)\, P_F(H' \mid H) = F(H')\, P_B(H \mid H')$ at every edge (Appendix D.1), and serves as a sanity check that the per-step $I(t)$ values are mutually consistent — a property that ablating P_ϕ (Table 2, −Backward policy) immediately breaks.

Step	Action	K_t	$\log \pi_\theta$	$\log P_\phi$	$I(t)$		$\log F(H_t)$
0	`[skill] tip-alfworld-routing`	–	0.00	0.00	1.00		-2.30
1	`[skill] tip-alfworld-place`	–	0.00	0.00	1.00		-2.30
2	go to shelf 1	5	-19.41	-12.50	0.001	$\diamond$	-9.21
3	go to dining table 1	6	-6.86	-14.20	1500	$\star$	-1.90
4	take watch 1	9	-5.30	-14.05	6300	$\star\star$	$+6.85$
5	go to shelf 1	5	-4.79	-12.80	3000	$\star$	$+14.86$
6	move watch 1 to shelf 1	8	-5.30	-14.46	9500	$\star\star$	$+24.02$
7	go to dining table 1	6	-5.10	-13.91	6700	$\star$	$+32.83$
8	take watch 2	9	-5.92	-13.39	1750	$\star$	$+40.30$
9	move watch 2 to shelf 1, `accept`	8	-5.16	-12.81	2100	$\star$	$+47.95$

Table 5: Per-step decomposition of a successful ALFWorld trajectory. K_t: action-token count; $I(t) = \pi_\theta(a_t \mid r_t, H_{t-1})/P_\phi(a_t \mid H_{t-1} \oplus o_t^{\text{exec}}) = \exp(\log \pi_\theta - \log P_\phi)$; $\log F(H_t) = \log Z_\theta(q) + \sum_{t' \le t} \log I(t')$ (main-text Eq. 11). $\diamond$: confirmed step; $\star$/$\star\star$: critical / highly-critical decision. Cumulative log-flow at t=9 matches $-2.30 + \sum_{t'=1}^{9} \log I(t') = 47.95$ exactly.

Q.5 Multi-Trajectory Comparison: SWE-bench Code Generation

Table 6 contrasts four trajectories sampled from a single SWE-bench query (a `pydap` signed-bytes patch). The two successes (τ_1^+, τ_3^+) and two failures (τ_2^-, τ_4^-) reveal a clean structural difference: every successful trajectory contains at least two high-$I(t)$ `edit_file` steps, whereas failures stall in repeated `search_code` / `view_file` loops without ever issuing a successful edit.

Trajectory	$\tilde{R}$	$\Delta(\tau)/T$	T	Critical-step pattern
τ_1^+ (success)	0.84	-0.06	16	search, edit$\times$2, verify – balanced flow
τ_2^- (failure)	0.45	$+0.22$	16	search OK; *no `edit_file` ever invoked*
τ_3^+ (success)	0.86	-0.04	14	search, edit (`LINT_ERROR`) $\rightarrow$ view $\rightarrow$ re-edit (try-fail-fix)
τ_4^- (failure)	0.37	$+0.18$	17	two `search_code` loops, no resolution

Table 6: Four-trajectory comparison on a single SWE-bench query. Successful trajectories carry a balanced TTB residual (Δ/T near zero) and concentrate $|\log I(t)| > 1$ on `edit_file` / `verify` steps; failures show large positive residuals and place high importance on early-stage search/view actions that never lead to a code change. The critical-step pattern is the signal that Ψ uses to synthesise new code-generation tips at phase boundaries.

The reward gap between the success and failure clusters is roughly 0.4–0.5, and the failure trajectories' high importance on non-editing actions is exactly the "*where* are the gaps?" signal that drives Ψ to create a new `tip-code-generation` skill specifying the search$\rightarrow$edit$\rightarrow$verify ordering.

R Limitations

SkillFlow targets one specific question: *can flow-based training drive recursive skill evolution from end-to-end task feedback alone?* Within that scope, our results substantiate the three claims (TTB convergence, zero-cost per-step credit, plateau-driven curation). The method's effectiveness, however, inherits two properties of the underlying language model that are themselves the subject of separate research lines.

Reliance on long-context memory. Multi-turn orchestration grows the supervisor's input by the full history $H_t = H_{t-1} \oplus (r_t, a_t, o_t^{\text{exec}})$ at every step. SkillFlow therefore relies on the backbone's ability to attend to and faithfully use long histories—backbones with weaker long-context fidelity see the per-step credit signal degrade as trajectories grow, since the hindsight backward P_ϕ must condition on increasingly distant context. Improving long-context modeling itself is an active research line whose advances stack with our training recipe but lie outside its scope.

Reliance on backbone reasoning capacity. As shown in §5.4 (RQ3), SkillFlow lifts every backbone but cannot replace the underlying base capability—orchestration amplifies what the model already knows about decomposition and tool use rather than installing new reasoning skills. Where the bottleneck is core reasoning rather than orchestration, additional pretraining or distillation is required; this remains complementary to, but outside the scope of, the present paper.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Introduction (§1) list three contributions—reward-proportional TTB training, a hindsight backward policy with zero-cost per-step credit, and flow-driven recursive skill evolution—each formalised in §4 and validated by RQ1–RQ5 in §5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: An explicit Limitations appendix (Appendix R) discusses backbone-scale scope, the frozen-executor assumption, the reward-positivity / β tuning requirement, and library-size scaling.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: All three propositions (Prop. 1, Prop. 2, Prop. 3) state their assumptions inline; their full proofs appear in Appendices E, J, and K, with supporting derivations in Appendices A–D, F, H, and I.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: §5.1 reports the experimental setup; Appendices M, N, O, and L provide datasets, baselines, evaluation metrics, and backward-policy implementation details, while Appendix Q gives per-step traces that allow direct verification.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: An anonymised repository containing training/evaluation code, configuration files, and skill-library snapshots is available at `https://anonymous.4open.science/r/SkillFlow-E850`. All datasets used are public and cited in §5.1 and Appendix M.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: §5.1 states the backbone, baselines, and metric definitions, and Appendices M–O extend these with dataset, baseline, and metric details; full hyperparameters and optimizer settings appear in the supplementary code.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Tables 1 and 2 report mean $\pm$ standard deviation across multiple training runs; the variability source is randomness in initialisation and trajectory sampling.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Hardware specifications and the main-run wall-clock / GPU-hours / hyperparameters are reported in Appendix P.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses publicly released datasets and open-source LLMs, involves no human subjects, and complies with the NeurIPS Code of Ethics throughout.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: On the positive side, SkillFlow lowers the engineering and compute cost of agentic systems and may reduce data-collection burden by reusing distilled skills. On the negative side, it inherits the dual-use concerns of capable LLM-based agents, while introducing no novel attack vector beyond existing autonomous-agent stacks.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: SkillFlow is a training-time framework built on existing public LLMs and benchmarks; no new high-risk pretrained models or scraped datasets are released.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets and backbone models used are cited with their original publications in §5.1 and Appendix M; each asset is used in compliance with its publicly stated license terms.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The SkillFlow framework, evolved skill libraries, and example trajectories are released in the anonymised supplementary material with documentation covering training scripts, configuration, and per-skill metadata.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The study involves no crowdsourced annotation and no human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: No human-subjects research is conducted; IRB approval is therefore not required.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs are central to the methodology: the Supervisor and Executor are LLM-based components, with Qwen3.5-9B (LoRA-tuned) as the primary trainable backbone and frontier proprietary models as alternative backbones (§5.1, Appendix L). The forward and hindsight backward policies, the TTB objective, and skill creation all operate over LLM next-token distributions.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.